Computer Science 351

Universal Turing Machines

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #12

Goal for Today

- Introduction of Turing machines that receive encodings of *other* Turing machines as inputs — along with strings over the input alphabets for these machines — and simulate the execution of these machines on these inputs.
- These are generally called ***universal Turing machines.*** Unfortunately textbooks do not generally describe these in much detail, but they are important!

Once again, students will be expected to understand ***concepts*** introduced and ***results*** that are proved — but not the (potentially confusing) details of the proofs that are presented.

Universal Turing Machines

You may have heard it said that...

- It is possible to write a ***Java compiler*** as a *Java program.*
- It is possible to write a ***Python emulator*** as a *Python program.*

You will learn today, that is possible to design a ***deterministic Turing machine*** that is a “Turing machine emulator.”

A Turing machine that does this is called a ***universal Turing machine.***

Universal Turing Machines: Complications

Proving the existence of a "universal Turing machine" M_{UTM} is complicated by several things:

- We have to figure out how a description (or ***encoding***) of some other Turing machine, M, can be included as part of the ***input string*** that the "universal Turing machine" M_{UTM} has to process.
- The sizes of the input alphabet and tape alphabet for the "encoded Turing machine" M can both be much larger than M_{UTM}'s tape alphabet. M might also have lots more states than M_{UTM} does.

So... this lecture has to start with details about "encodings," so that you can see how these problems can be overcome.

Universal Turing Machines: Setting Things Up

Suppose, from now, on, that the “encoded (input) Turing machine” is

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

and that the “encoded input string” is a string $\omega \in \Sigma^{\star}$.

Assumption: $q_0 \neq q_{\text{accept}}$ and $q_0 \neq q_{\text{reject}}$.

- It is easy to modify any Turing machine, without changing the language that this machine recognizes (or decides, if it decides one), so that it satisfies this assumption.

Universal Turing Machines: An Input Alphabet

Let

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

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

Universal Turing Machines: What M_{UTM} Receives as Input

M_{UTM} should receive a string in $\Sigma_{\text{TM}}^{\star}$ as input that has the form

$$(\mu\,,\nu)$$

where

- μ is a string in $\Sigma_{\text{TM}}^{\star}$ encoding some deterministic Turing machine
$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$
- ν is a string in $\Sigma_{\text{TM}}^{\star}$ encoding some input string $\omega \in \Sigma^{\star}$ for M

Universal Turing Machines: What M_{UTM} Receives as Input

- Thus M_{UTM} should start by checking whether *its* input string starts with “`(`”, includes at least one “`,`” and ends with “`)`”— ***rejecting*** it, if this is not the case.
- The encoding ν of M’s input string does not include a “`,`” — so the encoding μ of M includes all symbols between the initial “`(`” and the ***final*** “`,`” in M_{UTM}’s input string.

Encoding M's States as Strings in $\Sigma_{\text{TM}}^{\star}$

Renaming states if necessary, we may assume that

$$Q = \{q_0, q_1, \ldots, q_k, q_{\text{accept}}, q_{\text{reject}}\}$$

for some integer $k \geq 0$ — so that $|Q| = k + 3$.

- For $0 \leq i \leq k$, state q_i will be encoded by the string $e(q_i) \in \Sigma_{\text{TM}}^{\star}$ 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.
- q_{accept} will be encoded by the string $e(q_{\text{accept}}) =$ "`qY`".
- q_{reject} will be encoded by the string $e(q_{\text{reject}}) =$ "`qN`".
- ***Note:*** The encodings of the start state, accept state and reject state are all easy to recognize!

Encoding *M*'s Tape Symbols as Strings in $\Sigma_{\mathrm{TM}}^{\star}$

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

- $\sqcup$ will be encoded by a string $e(\sqcup) \in \Sigma_{\mathrm{TM}}^{\star}$ starting wth $\mathtt{s}$ and ending with ℓ $\mathtt{0}$'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

$$\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_h\}.$$

- For $1 \leq i \leq h$, σ_i will be encoded by the string $e(\sigma_i) \in \Sigma_{\mathrm{TM}}^{\star}$ starting with $\mathtt{s}$ and ending with a ***padded*** decimal representation of i — that is, a representation of i with enough leading $\mathtt{0}$'s to make sure that it is a string with length ℓ.

Encoding *M*'s Tape Symbols as Strings in $\Sigma_{\text{TM}}^{\star}$

- 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 $\sigma_{h+1}, \sigma_{h+2}, \dots, \sigma_m$ — 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 $\mathtt{s}$ and ending with the padded decimal representation of the number i with length ℓ
- ***Note:*** This ensures that ***every*** symbol in Σ is encoded by a string in $\Sigma_{\text{TM}}^{\star}$ 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 is also 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.

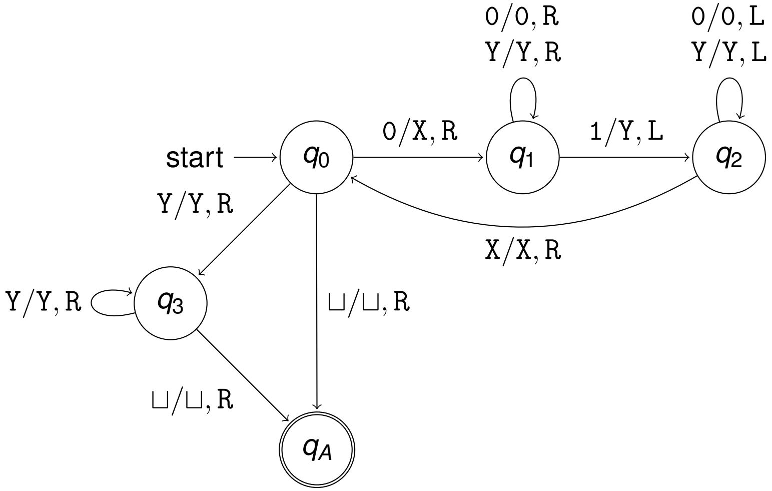
An Ongoing Example

The first example of a Turing machine provided in class — which decides the language

$$L = \{\mathtt{0}^n\mathtt{1}^n \mid n \geq 0\} \subseteq \Sigma^\star$$

over the alphabet $\Sigma = \{\mathtt{0}, \mathtt{1}\}$ — will be used as an ongoing example. The state diagram for this machine (as usual, leaving out the rejecting state and transitions to it) is as shown on the following slide.

An Ongoing Example



An Ongoing Example

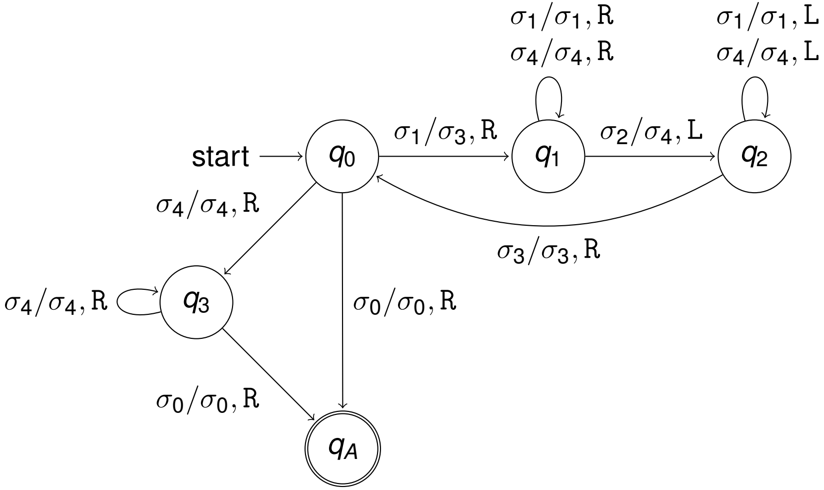
The states are already named as described above — with $k = 3$. Now

- $e(q_0) = \text{“q0”}$
- $e(q_1) = \text{“q1”}$
- $e(q_2) = \text{“q2”}$
- $e(q_3) = \text{“q3”}$
- $e(q_{\text{accept}}) = \text{“qY”}$
- $e(q_{\text{reject}}) = \text{“qR”}$

An Ongoing Example

- Choosing an ordering for the symbols in Σ, let us set $\sigma_1 = \mathtt{0}$ and $\sigma_2 = \mathtt{1}$.
- Choosing an ordering for the symbols in Γ that are not blank and are not in Σ, let us set $\sigma_3 = \mathtt{X}$ and $\sigma_4 = \mathtt{Y}$.
- A modified state diagram that uses these names for symbols is as follows.

Modified Example


start
q_0
q_1
q_2
q_3
q_A
$\sigma_1/\sigma_3, \mathrm{R}$
$\sigma_2/\sigma_4, \mathrm{L}$
$\sigma_1/\sigma_1, \mathrm{R}$
$\sigma_4/\sigma_4, \mathrm{R}$
$\sigma_1/\sigma_1, \mathrm{L}$
$\sigma_4/\sigma_4, \mathrm{L}$
$\sigma_3/\sigma_3, \mathrm{R}$
$\sigma_4/\sigma_4, \mathrm{R}$
$\sigma_4/\sigma_4, \mathrm{R}$
$\sigma_0/\sigma_0, \mathrm{R}$
$\sigma_0/\sigma_0, \mathrm{R}$

Modified Example

The transition table for this modified example Turing machine is as follows. In this table q_A is shown instead of q_{accept}, and q_R is shown instead of q_{reject}, so that everything fits into the slide.

	σ_1	σ_2	σ_0	σ_3	σ_4
q_0	$(q_1, \sigma_3, \mathtt{R})$	$(q_R, \sigma_2, \mathtt{R})$	$(q_A, \sigma_0, \mathtt{R})$	$(q_R, \sigma_3, \mathtt{R})$	$(q_3, \sigma_4, \mathtt{R})$
q_1	$(q_1, \sigma_1, \mathtt{R})$	$(q_2, \sigma_4, \mathtt{L})$	$(q_R, \sigma_0, \mathtt{R})$	$(q_R, \sigma_3, \mathtt{R})$	$(q_1, \sigma_4, \mathtt{R})$
q_2	$(q_2, \sigma_1, \mathtt{L})$	$(q_R, \sigma_2, \mathtt{R})$	$(q_R, \sigma_0, \mathtt{R})$	$(q_0, \sigma_3, \mathtt{R})$	$(q_2, \sigma_4, \mathtt{L})$
q_3	$(q_R, \sigma_1, \mathtt{R})$	$(q_R, \sigma_2, \mathtt{R})$	$(q_A, \sigma_0, \mathtt{R})$	$(q_R, \sigma_3, \mathtt{R})$	$(q_3, \sigma_4, \mathtt{R}$

Encoding M's Tape Symbols as Strings in $\Sigma_{\mathrm{TM}}^{\star}$

Now, since $|\Gamma| = 5$, $\ell = 1$ (since $10^0 < 5 \leq 10^1$), so that

- $e(\sqcup) = e(\sigma_0) =$ “$\mathtt{s0}$”
- $e(\mathtt{0}) = e(\sigma_1) =$ “$\mathtt{s1}$”
- $e(\mathtt{1}) = e(\sigma_2) =$ “$\mathtt{s2}$”
- $e(\mathtt{X}) = e(\sigma_3) =$ “$\mathtt{s3}$”
- $e(\mathtt{Y}) = e(\sigma_4) =$ “$\mathtt{s4}$”

Encoding Transitions as Strings in $\Sigma_{\text{TM}}^{\star}$

Each ***transition*** of M now has the form $\delta(q, \sigma) = (r, \tau, D)$ where

- $q \in \{q_0, q_1, q_2, \dots, q_k\}$,
- $\sigma \in \Gamma = \{\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_m\}$,
- $r \in Q = \{q_0, q_1, q_2, \dots, q_k, q_{\text{accept}}, q_{\text{reject}}\}$,
- $\tau \in \Gamma$, and
- $D \in \{\mathtt{L}, \mathtt{R}\}$.

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

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

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

Encoding Transitions as Strings in $\Sigma^{\star}_{\mathrm{TM}}$

- For example, the transition

$$\delta(q_0, \sigma_1) = (q_1, \sigma_3, R)$$

in the ongoing example is encoded by the string

$$e(\delta(q_0, \sigma_1)) = \text{“}\texttt{(q0,s1,q1,s3,R)}\text{”}$$

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 theTransition Function as a String in $\Sigma_{\text{TM}}^{\star}$

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

- starts with "("
- continues with the encodings of all the transitions $\delta(q, \sigma)$, separated by ","'s and
- ends with ")"

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

- If $0 \leq i < j \leq k$ 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 $0 \leq i \leq k$, if $0 \leq h < j \leq m$ then the encoding of $\delta(q_i, \sigma_h)$ should be listed before the encoding of $\delta(q_i, \sigma_j)$.

Encoding the Transition Function as a String in $\Sigma_{\text{TM}}^{\star}$

Note: If decimal representations of k and m are available (and stored on separate tapes, making them easy to find), each of the following things is easy to do using a deterministic Turing machine:

- Decide whether a given string in $\Sigma_{\text{TM}}^{\star}$ *is* a (valid) encoding of some transition $\delta(q, \sigma)$.
- Decide whether a given string in $\Sigma_{\text{TM}}^{\star}$ *is* a (valid) encoding $e(\delta)$ of the transition function.

Suggested Exercises: Write down (reasonably high level) descriptions of Turing machines that can do these things — assuming (correctly) that it is easy to add one to the decimal representation of number, use their decimal representations to decide whether two numbers are equal, or whether one number is less than or equal to the other.

Encoding a Turing Machine as a String in $\Sigma_{\mathrm{TM}}^{\star}$

The Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{accept}}, q_{\mathrm{reject}})$ 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 k (such that $|Q| = k + 3$, 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 Turing Machine as a String in $\Sigma_{\mathsf{TM}}^{\star}$

This makes it easy for a deterministic Turing machine to *decide* whether a string $\mu \in \Sigma_{\mathsf{TM}}^{\star}$ *is* a valid encoding of a Turing machine — and to set up tapes for a simulation, it is:

1. ***Reject*** μ if it does not start with "`(`".
2. ***Reject*** μ if it does not continue with the unpadded decimal representation of some number $k \in \mathbb{N}$, followed by "`,`". Otherwise write the decimal representation of k onto the beginning of another tape, so it is easy to find later.
3. ***Reject*** μ if it does not continue with the unpadded decimal representation of some *positive* integer h, followed by "`,`". Otherwise write the decimal representation of h onto the beginning of *another* tape, it is easy to find later, as well.

Encoding a Turing Machine as a String in $\Sigma_{\mathsf{TM}}^{\star}$

4. ***Reject*** μ if it does not continue with the unpadded decimal representation of a number *m*, such that $h \leq m$, followed by another "$,$". Otherwise write the decimal representation of *m* onto *another* tape, so that it is also easy to find later on.
5. ***Reject*** μ if it does not end with "$)$". Otherwise set $\xi \in \Sigma_{\mathsf{TM}}^{\star}$ to be the substring of μ starting immediately *after* the "$,$" mentioned in step 4, and immediately *before* the final "$)$".
6. ***Accept*** μ if ξ is a valid encoding of a transition function (consistent with the numbers *k*, ℓ and *m* described above). Otherwise ***reject*** μ.

The final step in this algorithm is tricky! Additional details about this are included in the lecture presentation.

Encoding an Input String as a String in $\Sigma_{\text{TM}}^{\star}$

A string $\nu \in \Sigma_{\text{TM}}^{\star}$ is an encoding of a string

$$\omega = \tau_1 \tau_2 \ldots \tau_n \in \Sigma^{\star}$$

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

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

Since the size, h of Σ is now available, it is easy to decide whether a given string $\nu \in \Sigma_{\text{TM}}^{\star}$ has this form, for some string $\omega \in \Sigma^{\star}$.

Encoding an Input String as a String in $\Sigma_{\mathsf{TM}}^{\star}$

M_{UTM} should ***reject*** its input string if it does not have the form (μ, ν) where μ is an encoding of some deterministic Turing machine

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

and where ν is an encoding of some string $\omega \in \Sigma^{\star}$. Otherwise M_{UTM} should simulate the execution of M on input ω, as described next.

Completing the Initialization

Three more tapes will be used:

- A tape, “Tape A,” that stores the encoding of the *current state* of M. This should be initialized to store the encoding q0 of q_0.
- A tape, “Tape B,” that represents the non-blank part of M’s tape. This should be initialized to store the string #ν, where ν is the encoding of the input string ω that is part of M_{UTM}’s input. The tape head should be moved back to the leftmost symbol in ν.

  Note: The only $\#$ on this tape is the one on the leftmost cell.
- A tape, “Tape C, can be used to store values needed for calculations. This can be left blank to start.

What M_{UTM} Does

M_{UTM} now simulates M, one move at a time, in a series of ***rounds***. At the beginning of each round,

- Tape A stores the current state of M.
- Tape B stores an encoding of the non-blank part of M's tape, with the tape head resting on the leftmost symbol in the encoding of the symbol (in Σ) that is currently visible on M's tape.
- The input tape has not been changed, so it still includes an encoding of the transition function for M.

Note: All these properties initially hold.

What M_{UTM} Does

M_{UTM} should continue as follows.

1. Sweep over the encoding of the transition function, δ, until the beginning of the encoding of transitions for the current state, q, are found — using Tape A to check for this.
2. Continue to sweep over the encoding of the transition function, δ, until the transition for q and the symbol σ, currently visible on M's tape, is found — using tape B to check for *this*.

Suppose that the transition found is

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

where $r \in Q$, $\tau \in \Gamma$, and $D \in \{\mathtt{L}, \mathtt{R}\}$.

What M_{UTM} Does

3. If r is q_{accept} then M_{UTM} should ***accept***. If r is q_{reject} then M_{UTM} should ***reject***. Otherwise, M_{UTM} should continue with the next step.
4. Replace the encoding of q with the encoding of r on Tape A.
5. Replace the (currently visible) encoding of σ with an encoding of τ on Tape B, moving the tape head back to the first symbol in this encoding of τ.
6. If $D = \mathtt{L}$ then move the tape head for Tape B to the beginning of the encoding of a symbol that is immediately to the left of the encoding of τ — unless $\#$ appears immediately to the left. Move the tape head back to the first symbol in the encoding of τ, in that case.

What M_{UTM} Does

7. If $D = \mathtt{R}$ then move the tape head for Tape B to the cell immediately to the right of the last symbol in the encoding of τ.

   If another $\mathtt{s}$ is now visible, leave the tape head where it is.

   Otherwise, $\sqcup$ is now visible. Replace this (and other $\sqcup$'s immediately to right with a copy of the *encoding* of $\sqcup$ and move the tape head back to the beginning of this encoding.

I hope it is not hard to see that another move of M has been simulated — and that the properties that should hold at the beginning of each round hold, once again.

Conclusions

Claim #1: If the universal Turing machine M_{UTM} that has now been described is executed with an input string $\mu \in \Sigma_{\text{TM}}^{\star}$, then:

- If μ does not encode a Turing machine M (with input alphabet Σ) and string $\omega \in \Sigma^{\star}$ at all, then M_{UTM} ***rejects*** μ.
- If μ encodes a Turing machine M and string $\omega \in \Sigma^{\star}$ such that M ***accepts*** ω then M_{UTM} ***accepts*** μ.
- If μ encodes a Turing machine M and string $\omega \in \Sigma^{\star}$ such that M ***rejects*** ω then M_{UTM} ***rejects*** μ.
- If μ encodes a Turing machine M and string $\omega \in \Sigma^{\star}$ such that M ***loops on*** ω then M_{UTM} ***loops on*** μ.

Conclusions

The most expensive of the simulation is “discovering the right transition to apply.” With a bit of work one can show that the number of steps needed for this is at most linear in the length of the encoding of the transition function — which is at most the length of the encoding of *M*.

This can be used to prove the following.

Claim #2: If M_{UTM} is executed on a string $\xi \in \Sigma_{\mathrm{TM}}^{\star}$ that encodes a Turing machine M and string $\omega \in \Sigma^{\star}$, then the number of steps used by M_{UTM} to simulate the first t steps of M on input ω is at most linear in the product of t and the length of the input string ξ.

Conclusions

The important result that has now been proved is as follows:

Claim #3: For every alphabet Σ, the language

$$\mathsf{A}_{\mathsf{TM}} \subseteq \Sigma_{\mathsf{TM}}^{\star}$$

consisting of strings μ encoding Turing machines *M* with some input alphabet Σ, and strings $\omega \in \Sigma^{\star}$, such that *M* accepts ω, is ***Turing-recognizable***.

By the end of the course, you will see a proof that this language is provably ***not*** Turing-decidable!