Computer Science 351

Universal Turing Machines and First Hard Problems

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #15

Goal for Today

- Introduction of ***universal Turing machines***.
- Identification of a language that is provably ***undecidable***.
- Identification of a language that is provably ***unrecognizable***.

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

- The input for a universal Turing machine must include a representation — called an ***encoding*** of some other Turing machine

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

  — as a *string of symbols* over the universal Turing machine's input alphabet.
- ***New Assumption:*** In the encoded Turing machine, $q_0 \neq q_{\text{accept}}$ and $q_0 \neq q_{\text{reject}}$.

The Universal Turing Machine's Input Alphabet

- Let

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

  — so that this alphabet includes twenty symbols, including a comma ("`,`") and semi-colon ("`;`").
- Σ_{TM} will be the ***input alphabet*** for the universal Turing machine being described.

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

Renaming states in M 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 $\mathtt{q}$ followed by the ***unpadded*** decimal representation of the number i — so that $e(q_0) =$ "$\mathtt{q0}$", $e(q_1) =$ "$\mathtt{q1}$", $e(q_{12}) =$ "$\mathtt{q12}$", and so on.
- q_{accept} will be encoded by the string $e(q_{\text{accept}}) =$ "$\mathtt{qY}$".
- q_{reject} will be encoded by the string $e(q_{\text{reject}}) =$ "$\mathtt{qN}$".

Encoding Tape Symbols

Let $\ell = |\Sigma|$ and let $m = |\Gamma|$, so that $m > \ell > 0$. Suppose that we ***enumerate*** the symbols in Γ as follows:

- σ_0 is the blank symbol, $\sqcup$;
- $\sigma_1, \sigma_2, \ldots, \sigma_\ell$ are the symbols in the input alphabet Σ, listed in some (fixed but arbitrary) order;
- $\sigma_{\ell+1}, \sigma_{\ell+2}, \ldots, \sigma_{m-1}$ are the remaining symbols in $\Gamma \setminus (\Sigma \cup \{\sqcup\})$, listed in some (fixed but arbitrary) order.
- Encode each symbol $\sigma_i \in \Gamma$ by a string, $e(\sigma_i)$, consisting of "s" followed by the unpadded decimal representation of i.

Encoding Directions of Motions

- Note that every direction of motion has a symbol in Σ_{TM} that represents it.
- We may therefore set $e(\mathtt{L}) = \mathtt{L}$ and $e(\mathtt{R}) = \mathtt{R}$ in order to ensure that each direction of motion is encoded by a string in $\Sigma_{\text{TM}}^{\star}$ with length one.

Encoding Transitions

Each "transition" is a rule with the form

$$\delta(q, \tau) = (\widehat{q}, \widehat{\tau}, d),$$

where $q, \widehat{q} \in Q$, and where $\tau, \widehat{\tau} \in \Gamma$ and $d \in \{\mathtt{L}, \mathtt{R}\}$.

- The above transition can now be encoded by a string $\zeta \in \Sigma_{\mathsf{TM}}^{\star}$ consisting of the strings

  $$e(q), e(\tau), e(\widehat{q}), e(\widehat{\tau}), e(d)$$

  separated by commas and enclosed by brackets "(" and ")".

Encoding Transition Functions

- A "transition function"

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$$

  can now be encoded as the encodings of the transitions

$$\delta(q, \tau)$$

  (such that $q \in Q$ and $\tau \in \Gamma$) that are defined — separated by commas and enclosed by left and right brackets.

- We will require that the encodings of transitions included here are ***sorted*** — so that all that transitions for start state $q = q_0$ appear before start state $q = q_1$, etc. Transitions for the same start state are sorted by τ.

Encoding Turing Machines

M can be encoded using the encodings of the following — separated by commas, and enclosed by brackets:

1. The ***unpadded decimal representation*** of the number $s = |Q| - 3$;
2. The ***unpadded decimal representation*** of the size $\ell = |\Sigma|$ of the input alphabet of M;
3. The ***unpadded decimal representation*** of the size $m = |\Gamma|$ of the tape alphabet of M;
4. The encoding of the transition function δ of M, as described above.

This defines a "standard encoding" $e(M)$ of M.

Definition: Let $L_{\mathsf{TM}} \subseteq \Sigma^{\star}_{\mathsf{TM}}$ be the language of encodings of Turing machines.

L_{TM} is Decidable

Claim #1: The language L_{TM} is decidable.

- Supplemental material for this lecture includes a sketch of a proof of this claim.

Encoding Turing Machines and Input Strings

- Consider a Turing machine M with input alphabet Σ. As above, let $\ell = |\Sigma|$, so that

$$\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_\ell\}.$$

- For a string

$$\omega = \alpha_1\alpha_2 \ldots \alpha_n \in \Sigma^\star$$

with length $n \geq 0$, encode ω as the string

$$e(\omega) = e(\alpha_1)e(\alpha_2) \ldots e(\alpha_n)$$

in $\Sigma^\star_{\mathsf{TM}}$.

Encoding Turing Machines and Input Strings

A Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ and an input string $\omega \in \Sigma^\star$ for M can now be encoded using the encoding of M and the encoding, $e(\omega)$, of ω, separated by a command and enclosed by brackets.

Definition: Let $L_{\text{TM+I}} \subseteq \Sigma^\star_{\text{TM}}$ be the language of encodings of Turing machines and input strings.

Claim #2: The language $L_{\text{TM+I}}$ is decidable.

- Supplemental material for this lecture includes a sketch of a proof of correctness of this claim.

Universal Turing Machines

A More Formal Definition:

Definition: A ***universal Turing machine*** is a (standard or multi-tape) Turing machine M_{UTM}, with input alphabet Σ_{TM}, such that, for every string $\mu \in \Sigma_{\text{TM}}^{\star}$, the following properties are satisfied:

- If $\mu \notin L_{\text{TM+I}}$ then M_{UTM} ***rejects*** μ.

Otherwise, $\mu \in L_{\text{TM+I}}$, so that μ encodes a Turing machine M (with some input alphabet Σ) and a string $\omega \in \Sigma^{\star}$.

- If M ***accepts*** ω then M_{UTM} ***accepts*** μ.
- If M ***rejects*** ω then M_{UTM} ***rejects*** μ.
- If M ***loops*** on ω then M_{UTM} ***loops*** on μ.

Universal Turing Machines

- An ***multi-tape Turing machine*** M_{UTM} (with three tapes) is described in supplemental material.
- M_{UTM} begins by checking whether its input string, $\mu \in \Sigma^{\star}_{\text{TM}}$, belongs to $L_{\text{TM+I}}$ — ***rejecting*** μ if $\mu \notin L_{\text{TM+I}}$.
- Otherwise, μ is an encoding of some (standard) Turing machine

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

  and a string $\omega \in \Sigma^{\star}$.
- M_{UTM} continues by ***simulating*** the execution of M on ω.

Universal Turing Machines

- The input string, μ, is kept on the first tape throughout the simulation — and is used to access ***M's transition function*** after the simulation begins.
- A second tape stores an ***encoding of a configuration of M***.
- During the simulation's ***Initialization Phase***, the encoding of *M*'s initial configuration for the given string, ω, is written onto the second tape.
- During the ***Step-by-Step Simulation*** information on the second tape is compared to the encoding of the transition function, to find the transition to be applied. This is then used to update the encoding of a configuration on the second tape.

Universal Turing Machines

- The simulation ends if — and when — it is discovered that M would enter its accepting or rejecting state; μ is accepted (respectively, rejected) by M_{UTM} when it has been discovered that ω would be accepted (respectively, rejected) by M.
- See supplemental material for additional details and a sketch of a proof of correctness of M_{UTM}.

An "Acceptance" Language

Definition: Let $A_{\text{TM}} \subseteq L_{\text{TM+I}}$ be the set of encodings of Turing machines

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

and strings $\omega \in \Sigma^\star$ such that ***M accepts*** ω.

Claim #3: The language A_{TM} is ***recognizable***.

Proof: $A_{\text{TM}} = L(A_{\text{TM}})$, for the universal Turing machine M_{UTM} described above. $\square$

A_{TM} is Undecidable

Claim #4: The language A_{TM} is ***undecidable***.

Proof: By contradiction.

- ***Assume*** that A_{TM} is decidable, so that there exists a Turing machine M_{ATM}, with input alphabet Σ_{TM}, such that M_{ATM} decides A_{TM}.
- Consider a Turing machine M_D, with input alphabet Σ_{TM}, which implements the algorithm on the following slide.

A_{TM} is Undecidable

Algorithm Implemented by M_D

On input $\mu \in \Sigma^{\star}_{TM}$:

1. `if` ($\mu \in L_{TM}$ and encodes a Turing machine M_μ with input alphabet Σ_{TM}) $\{$
2. `if` (M_μ accepts μ) $\{$
3. `reject` μ
   $\}$ `else` $\{$
4. `accept` μ
   $\}$
   $\}$ `else` $\{$
5. `reject` μ
   $\}$

A_{TM} is Undecidable

- Since L_{TM} is decidable, the test whether $\mu \in L_{\mathrm{TM}}$, at step 1, can be implemented.
- It is easy to check whether the implemented Turing machine has an input alphabet with size $|\Sigma_{\mathrm{TM}}| = 20$ — which is all that is needed to complete the test at step 1.
- ***By assumption*** A_{TM} is decidable — so the test at step 2 can be implemented too.
- So there really *is* a Turing machine, M_D that implements this algorithm.
- Let $\mu_D \in \Sigma^{\star}_{\mathrm{TM}}$ be the encoding of M_D.

A_{TM} is Undecidable

Question: What does M_D when executed on its own encoding?

- Tracing the execution of the algorithm one can confirm that

$$M_D \text{ accepts } \mu_D \iff M_D \text{ rejects } \mu_D.$$

- This is impossible, so a ***contradiction*** has been obtained.
- The only assumption made must be false: A_{TM} is undecidable. $\square$

The Complement of A_{TM} is Unrecognizable

Now consider the language

$$A_{\mathrm{TM}}^{C} = \{\mu \in \Sigma_{\mathrm{TM}}^{\star} \mid \mu \notin A_{\mathrm{TM}}\}.$$

Claim #5: The language A_{TM}^{C} is ***unrecognizable***.

Proof: By contradiction.

- ***Assume*** that A_{TM}^{C} is recognizable, so there exists a Turing machine M_N, with input alphabet Σ_{TM}, such that $L(M_N) = A_{\mathrm{TM}}^{C}$.
- By Claim #3 A_{TM} is recognizable, so there also exists a Turing machine M_Y, with input alphabet Σ_{TM}, such that $L(M_Y) = A_{\mathrm{TM}}$.

The Complement of A_{TM} is Unrecognizable

- Consider a multi-tape Turing machine M_D, with input alphabet Σ_{TM}, that simulates *both* M_Y and M_N ***in parallel:*** Simulating the first step of both machines, the first *two* steps of both machines, and so on.
- ***At least one of these simulations must halt.***
- At this point, M_D can accept its input string μ if $\mu \in A_{TM}$, and it can reject its input string if $\mu \notin ATM$ (that is, if $\mu \in A_{TM}^C$).
- That is, $\boldsymbol{M_D}$ ***decides*** $\boldsymbol{A_{TM}}$, so A_{TM} must be ***decidable.***
- However, Claim #4 implies that A_{TM} is ***undecidable*** — a ***contradiction*** has been obtained.
- So the assumption must be false: A_{TM}^C is ***unrecognizable***.

□

Wrapping Things Up

The instructor will now discuss...

- the significance of these results,
- what is expected of students concerning them,
- what one might find in other references, and
- what is coming next.