CS310 Automata Theory – 2017-2018

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Module 3: Turing machines, computability

Last two modules

Regular languages, NFA/DFA, Regular expressions, Myhill-Nerode relations.

2DFA: DFA + two-way head movement. They recognize exactly regular languages.

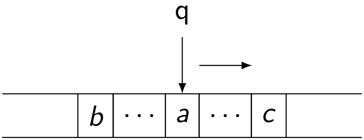
Finite state transducers (FSTs). Machines that output languages.

Pushdown automata: NFA + Stack. The class of languages recognized by these is called Context-free languages (CFLs).

Context-free grammars: Recursive programs. The class of languages generated by these grammars is CFLs.

Turing machines

What is a Turing machine? (Informal description.)



Read and write on the input tape. Head moves left/right.

The tape is infinite.

A special symbol & to indicate blank cells.

Initially all cells blank except the part where the input is written.

Special states for accepting and rejecting.

Formal definition

Definition

A Turing machine (TM) is given by $M = (Q, \Sigma, \Gamma, \delta, q_0, q_f, q_{rej})$

Q:	set of states	Σ:	input alphabet
q_0:	start state	Γ:	tape alphabet, $\Sigma \subseteq \Gamma$, $\& \in \Gamma$
q_{acc}:	accept state	q_{rej}:	reject state

$\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$.

Understanding δ

For a $q \in Q, a \in \Gamma$ if $\delta(q, a) = (p, b, L)$,

then p is the new state of the machine,

b is the letter with which a gets overwritten,

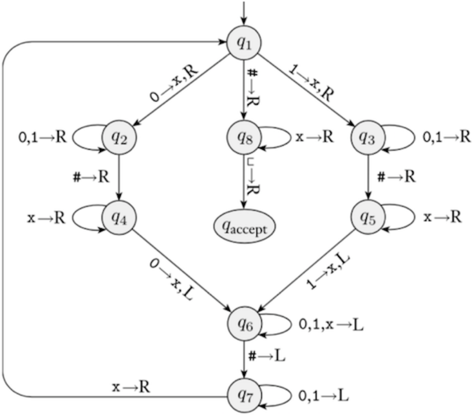
the head moves to the left of the current position.

Turing machine for a non-context free language

Example

$$\mathrm{EQ} = \{w \cdot \# \cdot w \mid w \in \Sigma^*\}.$$

Example from Sipser



Another example from Sipser

$$L = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}.$$

M_3 = "On input string w:

1. Scan the input from left to right to determine whether it is a member of $a^+b^+c^+$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and the c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*.
4. Restore the crossed off b's and repeat stage 3 if there is another a to cross off. If all a's have been crossed off, determine whether all c's also have been crossed off. If yes, *accept*; otherwise, *reject*."

Configuration

Definition

The configuration of a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_f, q_{rej})$ is given by

$$\Gamma^* \times Q \times \Gamma^*$$

A configuration need not include blank symbols.

Let $u, v \in \Gamma^*$, $a, b, c \in \Gamma$ and $q, q' \in Q$.

Suppose $(q', c, L) \in \delta(q, b)$ is a transition in M,
then starting from $u \cdot a \cdot q \cdot b \cdot v$ in one step we get $u \cdot q' \cdot a \cdot c \cdot v$.

We say that $u \cdot a \cdot q \cdot b \cdot v$ **yields** $u \cdot q' \cdot a \cdot c \cdot v$.
We denote it by $u \cdot a \cdot q \cdot b \cdot v \mapsto u \cdot q' \cdot a \cdot c \cdot v$.

Special configurations

Start configuration

We assume that the head is on the left of the input in the beginning.

Therefore, $q_0 \cdot w$ is the start configuration.

Accepting configuration

Any configulation that contains q_{acc} is an accepting configuration.

Rejecting configuration

Any configulation that contains q_{rej} is a rejecting configuration.

Halting configurations: if a configuration is accepting or rejecting then it is called a halting configuration.

A TM may not halt!

Acceptance by a TM

A TM M is said to accept a word $w \in \Sigma^*$ if there exists a sequence of configurations $C_0, C_1, \ldots, C_k$ such that

- C_0 is a start configuration,
- $C_i \mapsto C_{i+1}$ for all $0 \le i \le k-1$,
- C_k is an accepting configuration.

Let $\rho = C_0, C_1, \ldots, C_k$ be a sequence of configuration of M on w.

- This sequence ρ is called a run of the machine M on w.
- If C_k is an accepting configuration then ρ is called an accepting run.
- If C_k is a rejecting configuration then ρ is called a rejecting run.

Turing recognizable languages

Definition

A language L is said to be Turing recognizable if there is a Turing machine M such that $\forall w \in L$, M has at least one accepting run on w.

We say that M recognizes L.

For words not in L

the machine may run forever,

or may reach q_{rej},

both are valid outcomes,

and the machine is allowed to do either of the two.

Turning decidable languages

Definition

A language L is said to be Turing decidable if there is a Turing machine M such that for all $w \in \Sigma^*$, M halts on w and

if $w \in L$, M has an accepting run on w.

if $w \notin L$, all runs of M on w are rejecting runs.

We say that M decides L.

If a language L is Turing decidable then

the TM deciding L always halts.

L is also Turing recognizable.

Turing decidable languages form a subclass of Turing recognizable languages.

Comparing decidability and recognizability

Theorem

A language L is Turing decidable if and only if L and $\overline{L}$ are both Turing recognizable.

Proof.

$(\Rightarrow)$

If L is Turing decidable then L is also Turing recognizable (as we just saw).

If L is Turing decidable, then $\overline{L}$ is also Turing decidable. (Needs proof.)

Therefore, $\overline{L}$ is also Turing recognizable.

$(\Leftarrow)$

Let M_1, M_2 be two TMs recognizing $L, \overline{L}$, respectively.

We wish to come up with a TM M that will decide L.

Idea: on input w run both M_1, M_2, if M_1 reaches accepting configuration then accept. Else M_2 will reach the accepting configuraion. In that case, reject.

Variants of Turing machines

k-tape Turing machines

Usual TM + Multiples tapes + independent tape-head for each tape.

$\delta \subseteq Q \times \Gamma^k \times Q \times \Gamma^k \times \{L, R, S\}^k$.

Example

Given:	1^n on the input tape
Output:	1^{n^2} on the same tape.

k-tape Turing machines

Example

Given: 1^n on the input tape

Output: 1^{n^2} on the same tape.

0 While there is a 1 symbol on the first tape,

  0.1 Change the leftmost 1 symbol to X.

  0.2 For each X or 1 symbol on the first tape

    write a 1 symbol on the second tape.

  end for

end while

1 Copy the contents of the second tape on the first tape.

2 Halt and accept.

Variants of Turing machines

k-tape Turing machines

Usual TM + Multiples tapes + independent tape-head for each tape.

$\delta \subseteq Q \times \Gamma^k \times Q \times \Gamma^k \times \{L, R, S\}^k$.

Example

Given:	1^n on the input tape
Output:	1^{n^2} on the same tape.

Are k-tape TMs more powerful than 1-tape TMs?

Theorem

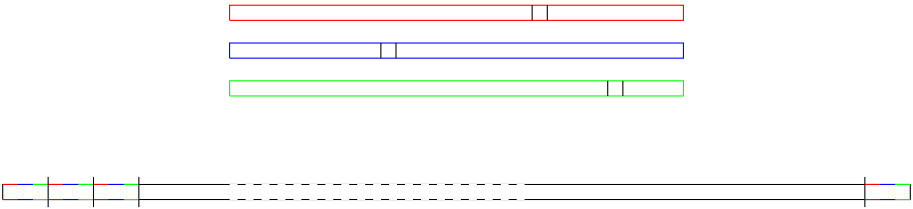
Every k-tape Turing machine has an equivalent 1-tape Turing machine.

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej},)$ be the k-tape Turing machine.

Let $M' = (Q', \Sigma, \Gamma', \delta', q_0, q_{acc}, q_{rej})$ be such that,

$\overline{\Gamma} = \{\overline{a} \mid a \in \Gamma\}$, $\Gamma = \Gamma \cup \overline{\Gamma} \cup \{\#\}$.

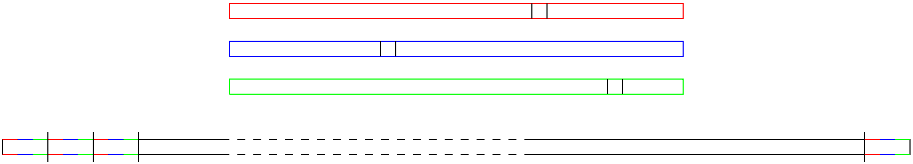
$\overline{\Gamma}$ symbols used to denote tape head positions.

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



To simulate 1 step of M, M' works follows:

reads the tape left to right once, remembeing the marked symbols in its states,

uses δ to determine the next state,

sweeps the input left to right again to update marked symbols.

Back to Comparing decidability and recognizability

Theorem

A language L is Turing decidable if and only if L and $\overline{L}$ are both Turing recognizable.

Proof.

$(\Rightarrow)$

If L is Turing decidable then L is also Turing recognizable

If L is Turing decidable, then $\overline{L}$ is also Turing decidable. (Needs proof.)

Therefore, $\overline{L}$ is also Turing recognizable.

$(\Leftarrow)$

Let M_1, M_2 be two TMs recognizing $L, \overline{L}$, respectively.

We wish to come up with a TM M that will decide L.

Idea: on input w run both M_1, M_2, if M_1 reaches accepting configuration then accept. Else M_2 will reach the accepting configuraion. In that case, reject.

□

Turing recognizability for $L, \overline{L} \Rightarrow$ Turing decidibility for L

We design 2-tape TM M, using TMs M_1, M_2 as follows:

M copies input from tape 1 to tape 2.

It acts as M_1 on tape 1 and as M_2 on tape 2.

M keeps track of the state control of M_1, M_2 in $Q_1 \times Q_2$.

Can you give a full decsription of M?

Turing machines as strings

Every TM can be represented as a string in $\{0,1\}^*$.

Just encode the description of the machine.

Every string over $\{0,1\}^*$ represents some TM.

If a string does not represent any TM, as per our encoding, let us assume that it represents a TM that does nothing.

Every TM is represented by infinitely many strings.

Any encoding of TMs will have a null character, say 010101. Then for any string $\alpha \in \{0,1\}^*$, suppose it represents machine M then all strings of the form $(010101)^*\alpha$ also represent the same machine M.

This has a similar effect as adding comments in the C program.

Notation

$M \longrightarrow \langle M \rangle$, a string representation of M.

$\alpha \longrightarrow M_\alpha$, a machine corresponding to α.

$\{0, 1\}^*$ is countable

Definition (Countable set)

A set S is said to be countable if there is an injective map from S to $\mathbb{N}$.

Lemma

The set $\{0, 1\}^*$ is countable.

Proof.

Let $x \in \{0, 1\}^*$.

Let $\phi(x)$ be defined as the number in $y \in \mathbb{N}$ such that y is a binary encoding of the number $1x$.

ϕ is a map from $\{0, 1\}^*$ to $\mathbb{N}$.

If $|x| \neq |x'|$ then $\phi(x) \neq \phi(x')$. If $|x| = |x'|$, then $\text{bin}(1x) \neq \text{bin}(1x')$ as long as $x \neq x'$.

Hence the map ϕ is injective.

Cantor's diagonalisation

Theorem (Cantor, 1891)

There is no bijection between $\mathbb{N}$ and $2^{\mathbb{N}}$ (set of all subsets of $\mathbb{N}$).

Proof.

Suppose for the sake of contradiction that there is a bijection, say f, between set of all subsets of $\mathbb{N}$.

	0	1	2	3	...
$\emptyset$	✗✓	✗	✗	✗	...
$\{1\}$	✗	✓ ✗	✗	✗	...
$\{2\}$	✗	✗	✓ ✗	✗	...
$\{1,2\}$	✗	✓	✓	✗ ✓	...
:	...	...	...	...	...
:	...	...	...	...	...

The inverted diagonal set does not belong to any of the existing sets! $\square$

Turing recognizable languages

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

Let L be a language, i.e. $L \subseteq \Sigma^*$, $w \in \Sigma^*$.

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise} \end{cases}$$

$$\text{languages over } \Sigma^* \xrightarrow{\text{bijection}} 2^{\mathbb{N}}$$

Therefore, set of all languages is uncountable.

However, the set of all TMs is countable. ($\{0,1\}^*$ is countable.)

There must be a language which is not Turing recognizable.

A decision problem about TMs

$$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$$

Lemma

A_{TM} is Turing recognizable.

Proof sketch

Design a TM, say N such that,

N behaves like M on w at each step,

if M reaches q_{acc} then N also accepts.

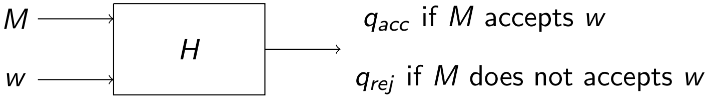
Is A_{TM} decidable?

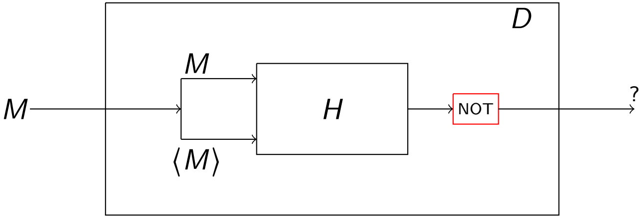
A decision problem about TMs

Lemma

$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$ is not Turing decidable.

Assume that there exists M such that M decides A_{TM}.



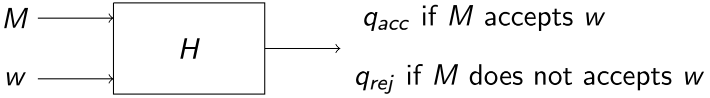


A decision problem about TMs

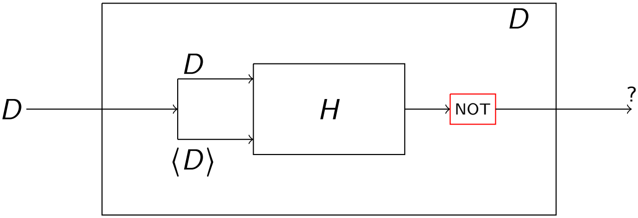
Lemma

A_{TM} is not Turing decidable.

Assume that there exists M such that M decides A_{TM}.



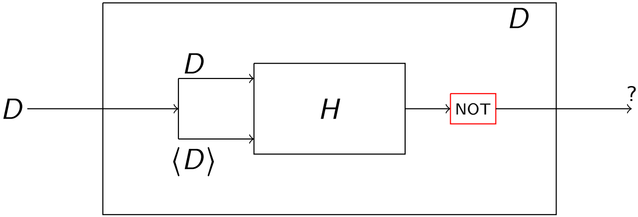
What happens if we give D as input to itself?



A decision problem about TMs

Lemma

A_{TM} is not Turing decidable.



If D accepts $\langle D\rangle$ then D rejects $\langle D\rangle$.

If D rejects $\langle D\rangle$ then D accepts $\langle D\rangle$. ☹

A few notable things

Note the following about the proof.

H accepts $\langle M, w \rangle$ when M accepts w.

D rejects $\langle M \rangle$ when M accepts $\langle M \rangle$.

D rejects $\langle D \rangle$ when D accepts $\langle D \rangle$.

Diagonalization inside the proof

Behaviour of the machines.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots\ldots$	$\ldots\ldots$
M_1	$\checkmark$		$\checkmark$	$\checkmark\ldots$	$\ldots\ldots$
M_2	$\checkmark$	$\times$		$\times\ \ldots$	$\checkmark\ldots\times\checkmark\ \ldots$
M_3	$\times$	$\times$	$\checkmark$	$\ldots\times$	$\checkmark\ldots\ldots$
$\vdots$					
$\vdots$					

Diagonalization inside the proof

Behaviour of H.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots\ldots$	$\ldots\ldots$
M_1	$\checkmark$	$\times$	$\checkmark$	$\checkmark\ldots$	$\ldots\ldots$
M_2	$\checkmark$	$\times$	$\times$	$\times\ \ldots$	$\checkmark\ldots\times\checkmark\ \ldots$
M_3	$\times$	$\times$	$\checkmark$	$\ldots\times$	$\checkmark\ldots\ldots$
$\vdots$					
$\vdots$					

Diagonalization inside the proof

Behaviour of H.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots\ldots$	$\ldots\ldots$
M_1	$\checkmark$	$\times$	$\checkmark$	$\checkmark\ldots$	$\ldots\ldots$
M_2	$\checkmark$	$\times$	$\times$	$\times\ \ldots$	$\checkmark\ldots\times\checkmark\ \ldots$
M_3	$\times$	$\times$	$\checkmark$	$\ldots\ \times$	$\checkmark\ldots\ldots$
$\vdots$					
$\vdots$					

Diagonalization inside the proof

Behaviour of D.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots\ldots$	$\ldots\ldots$
M_1	$\cancel{\checkmark}$ $\times$	$\times$	$\checkmark$	$\checkmark \ldots$	$\ldots\ldots$
M_2	$\checkmark$	$\cancel{\times}$ $\checkmark$	$\times$	$\times \ldots$	$\checkmark \ldots \times \checkmark \ldots$
M_3	$\times$	$\times$	$\cancel{\checkmark}$ $\times$	$\ldots \times$	$\checkmark \ldots\ldots$
$\vdots$					
$\vdots$					

Diagonalization inside the proof

Behaviour of D on itself.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots \langle D \rangle \ldots$	$\ldots\ldots$
M_1	$\cancel{\checkmark}$ $\times$	$\times$	$\checkmark$	$\checkmark \ldots$	$\ldots\ldots$
M_2	$\checkmark$	$\cancel{\times}$ $\checkmark$	$\times$	$\times \ \ldots$	$\checkmark \ldots \times \checkmark \ \ldots$
M_3	$\times$	$\times$	$\cancel{\checkmark}$ $\times$	$\ldots \ \times$	$\checkmark \ldots\ldots$
$\vdots$					
$\vdots$					
D				$\ldots \quad ? \ldots$	$\ldots\ldots$

Deteministic Turing machines

Theorem

Let L be a language decided by a non-deterministic TM N. Then there is a deterministic Turing machine M such that M decides L.

Possible proof idea:

- Think of the runs of N on an input w as a tree.
- Do DFS.
  - Simulate the non-deterministic TM one run at a time.
  - If the run accepts then accept and halt.
  - Else go to the next path.
- If after exploring all the paths we do not reach the accept state, then reject.

Corollary

If L is decidable then $\overline{L}$ is also decidable.

Deteministic Turing machines

Theorem

Let L be a language recognised by a non-deterministic TM N. Then there is a deterministic Turing machine M such that M recognises L.

Possible proof idea:

- Think of the runs of N on an input w as a tree.
- Do DFS.
  - Simulate the non-deterministic TM one run at a time.
  - If the run accepts then accept and halt. Else ... ?

DFS does not work! Use BFS. Proof Idea:

- Explore the tree of the NTM in rounds.
- In round i,
- for $1 \leq k \leq 2^i$
  - open up the kth runs of the NTM of length i.
  - if it is an accepting run then accept
  - else go to next k
- endfor

Deteministic Turing machines

Theorem

Let L be a language recognised by a non-deterministic TM N. Then there is a deterministic Turing machine M such that M recognises L.

For exploring the tree

Note that the degree of every node of the tree is at most

$$D := \max_{a \in \Gamma, q \in Q}\{|\delta(q, a)|\}$$

Hence, the tree is a D-ary tree.

For exploring runs of length i, keep track of the current run using a string over $[D]^i$.

Say $D = 3$, the kth path of length 5 is $1, 2, 3, 1, 2$ then the $k+1$th path of length 5 is $1, 2, 3, 1, 3$.

Back to Comparing decidability and recognizability

Theorem

A language L is Turing decidable if and only if L and $\overline{L}$ are both Turing recognizable.

Proof.

($\Rightarrow$)

If L is Turing decidable then L is also Turing recognizable

If L is Turing decidable, then $\overline{L}$ is also Turing decidable.

Therefore, $\overline{L}$ is also Turing recognizable.

($\Leftarrow$)

Let M_1, M_2 be two TMs recognizing $L, \overline{L}$, respectively.

We wish to come up with a TM M that will decide L.

Idea: on input w run both M_1, M_2, if M_1 reaches accepting configuration then accept.

Else M_2 will reach the accepting configuraion. In that case, reject.

$\square$

A decision problem about TMs

$$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$$

Lemma

A_{TM} is Turing recognizable.

Proof sketch

Design a TM, say N such that,

N behaves like M on w at each step,

if M reaches q_{acc} then N also accepts.

Is A_{TM} decidable?

Universal Turing machines

Definition

A Turing machine is called a Universal Turing machine if it can given the decsription of any Turing machine M and an input w, simulate the machine M on w.

Lemma

Universal Turing machine (UTM) exists. *[Turing, 1940s]*

Proof.

We will prove the lemma by explicitly constructing such a machine.

Proof idea:

Find a good encoding for Turing machines.
Tape 1: Hold the input, namely M and w.
Tape 2: Copy the decription of M and use it for referencing moves.
Tape 3: Store the current state M and letter of w being read.

□

Other undecidable problems and reducibility

Reducing A_{TM} to another problem to prove undecidibility.

$$\text{Halt} = \{(M, w) \mid M \text{ halts on } w\}$$

We would like to show that Halt is undecidable.

Assume that Halt is decidable. Let $\mathcal{H}$ be the TM deciding Halt.

$\mathcal{A}$: Run $\mathcal{H}$ on (M, w). If it rejects then reject, else do as per M on w.

$\mathcal{A}$ accepts (M, w) if M accepts w and rejects it if either M rejects w or M loops forever on w.

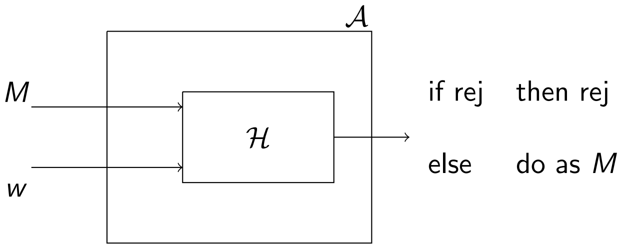
$\mathcal{H}$ decides Halt if and only if $\mathcal{A}$ decides A_{TM}.

The halting problem

Lemma

The halting problem, $Halt = \{(M, w) \mid M \text{ halts on } w\}$, is undecidable.

Another way to describe the same proof.



If Halt is decidable then $\mathcal{A}$ decides A_{TM}, which is a contradiction.

Emptiness problem for TM

Lemma

The emptiness problem for TMs, $E_{TM} = \{\langle M \rangle \mid L(M) = \varnothing\}$, is undecidable.

Assume for the sake of contradiction that it is decidable. Let T be a machine that decides E_{TM}.

Let $T'_{M,w}$ be as follows:

On input x

{

if $w \neq x$ then reject

else do as per M

}

$$L(T'_{M,w}) = \begin{cases} \{w\} & \text{if } M \text{ acc } w \\ \varnothing & \text{otherwise} \end{cases}$$

Let A be as follows:

On input M, w

{

Create machine $T'_{M,w}$.

If T on $\langle T'_{M,w} \rangle$ rejects

then accept

else reject

}

This shows that if E_{TM} is decidable then A_{TM} is decidable.

Equality for TM

Lemma

The equality problem for TMs, $EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}$, is undecidable.

Assume for the sake of contradiction that EQ_{TM} is decidable. Let M be the TM for it.

Let M_1 be a machine that rejects all strings. That is, $L(M_1) = \emptyset$.

Given a machine M_2 as an input, use M to check whether $L(M_2) = L(M_1)$, i.e. to check whether $L(M_2) = \emptyset$ or not.

This implies that if EQ_{TM} is decidable then E_{TM} is decidable.

But from the previous result we know that E_{TM} is undecidable.

Regularity checking

Lemma

$REG_{TM} = \{\langle M\rangle \mid L(M)$ is regular$\}$ is undecidable.

Assume for the sake of contradiction that a TM R is a TM that decides REG_{TM}.

Let $R'_{M,w}$ be s.t.

$$L(R'_{M,w}) = \begin{cases} \{0^n1^n \mid n \geq 0\} & \text{if } M \text{ rej } w \\ \Sigma^* & \text{if } M \text{ acc } w \end{cases}$$

If we get such an $R'_{M,w}$ we can design A as a follows.

Let A be as follows:

On input M, w

{

Create machine $R'_{M,w}$.

If R on $\langle R'_{M,w}\rangle$ accepts

then accept

else reject

}

Regularity checking

Lemma

$REG_{TM} = \{\langle M \rangle \mid L(M)$ is regular$\}$ is undecidable.

Assume for the sake of contradiction that a TM R be a TM that decides REG_{TM}.

Let $R'_{M,w}$ be as follows:

On input x

$\{$

if $x = 0^n 1^n$

then accept

else run M on w and

if M acc w then acc

else rej

$\}$

Let A be as follows:

On input M, w

$\{$

Create machine $R'_{M,w}$.

If R on $\langle R'_{M,w} \rangle$ accepts

then accept

else reject

$\}$

Rice's theorem

The following languages are undecidable.

$\{M \mid L(M) \text{ is regular}\}$.

$\{M \mid L(M) \text{ is context-free}\}$.

$\{M \mid L(M) = \varnothing\}$.

The following languages are decidable.

$\{M \mid M \text{ has more than 10 states}\}$.

$\{M \mid M \text{ does not have a left move}\}$.

Rice's theorem: A systematic way of proving undecidability of languages.

Property P

Definition

A property P is simply a subset of Turing recognizable languages. We say that a language L satisfies a property P, if $L \in P$.

Examples

Set of regular languages. ✓

Set of context-free languages. ✓

$\{\varnothing\}$. ✓

Rice's theorem

Definition

A property P of Turing recognizable languages is called a non-trivial property if

> there exists a TM M such that $L(M) \in P$, and
>
> there exists a TM M' such that $L(M') \notin P$.

Examples

> Set of all TMs whose language is Σ^* ✓
>
> Set of all Turing recognizable languages such that a TM recognising it has at least 10 states. $\times$

Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Property P and $\mathcal{L}_P$

Definition

A property P is simply a subset of Turing recognizable languages. We say that a language L satisfies a property P, if $L \in P$.

For any property P, let $\mathcal{L}_P = \{M \mid L(M) \in P\}$, i.e. the set of all Turing machine such that $L(M) \in P$.

We say that a property P is trivial if either $\mathcal{L}_P = \varnothing$ or $\mathcal{L}_P$ is the set of all the Turing recognizable languages.

Examples of properties

Examples

1. $\mathcal{L}_P = \{M \mid L(M) \text{ is regular}\}$.

   $\mathcal{L}_P$ is collection of TMs M such that $L(M)$ is regular.

   Is $\mathcal{L}_P = \varnothing$? No. For example, a TM accepting $a^* b^*$ is in $\mathcal{L}_P$.

   Is $\mathcal{L}_P =$ all TMs? No. For example, a TM accepting $\{a^n b^n \mid n \geq 0\}$ is not in $\mathcal{L}_P$.

   Therefore, P is not trivial.

Examples of properties

Examples

2 $\mathcal{L}_P = \{M \mid L(M) = \varnothing\}$.

Here $\mathcal{L}_P$ is a collection of TMs M such that $L(M) = \varnothing$.

Is $\mathcal{L}_P = \varnothing$? No. For example, a TM M that rejects any string is in $\mathcal{L}_P$.

Is $\mathcal{L}_P =$ all TMs? No. For example, a TM M that accepts a single string $\{a\}$ is not in $\mathcal{L}_P$.

Example of a trivial property

Examples

3. $\mathcal{L}_P = \left\{ M \;\middle|\; \begin{array}{c} M \text{ is a TM and } L(M) \text{ is accepted by} \\ \text{a TM that has even number of states} \end{array} \right\}.$

Here P is a property of Turing recognizable languages.

But any TM can be converted into another one that has even number of states.

Therefore, any Turing recognizable language has property P.

Therefore, P is in fact all Turing recognizable languages.

Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

When is the theorem NOT applicable?

- When P is a property about TMs and not about Turing recognizable languages.
  - $\{\langle M\rangle \mid M$ has at least ten states$\}$.
  - $\{\langle M\rangle \mid M$ never moves left on any input string $\}$.
  - $\{\langle M\rangle \mid M$ has no useless state $\}$.
- To prove non-recognizability of a property of languages.
  - Rice's theorem cannot be used to prove non-recognizability of languages.
  - It is only used to prove undecidability.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{\langle M \rangle \mid M$ runs for atmost 10 steps on $aab\}$.

Not applicable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$$\{\langle M\rangle \mid L(M) \text{ is recognized by a TM with at least 10 states}\}.$$

Applicable, but property is trivial.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$$\{\langle M\rangle \mid L(M) \text{ is recognized by a TM with atmost 10 states}\}.$$

Applicable and property is not trivial, therefore undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{\langle M\rangle \mid M$ has at most 10 states$\}$.

Not applicable, but the language is decidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{M \mid L(M) \text{ contains } \langle M \rangle\}$.

Applicable, the property is not trivial, therefore undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{M \mid L(M) \text{ contains } \langle M \rangle\}$.

Applicable, the property is not trivial, therefore undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$$\{\langle M\rangle \mid L(M) \text{ is recognized a TM with atmost 10 states}\}.$$

Applicable.
If we can simulate any TM with another with less than 10 states, then the property will be trivial.

This is doable if we allow for the tape alphabet size to grow.

In that case, the property is trivial.

Textbooks usually consider this property to be not trivial.

This is because the usual assumption is that you always fix the tape alphabet.
In that case, Rice's theorem is applicable and the property is not trivial, therefore undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{\langle M \rangle \mid M \text{ has at most 10 states}\}$.

Not applicable, but the language is decidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{M \mid L(M) \text{ contains } \langle M \rangle\}$.

Applicable, the property is not trivial, therefore undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{M \mid L(M) \text{ contains } \langle M \rangle\}$.

Applicable, the property is not trivial, therefore undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{M \mid L(M) \text{ is finite }\}$.

Applicable, the property is not trivial, therefore undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$$\{M \mid L(M) = \Sigma^*\}.$$

Applicable, the property is not trivial, therefore undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{(M, w) \mid M$ writes a symbol a on the tape on input $w\}$.

Not applicable, but the language is in fact undecidable.

Rice's theorem cannot be used to prove the undecidability of this language!

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$$\left\{ M \middle| \begin{array}{l} M \text{ tries to write on the left of the cell when it} \\ \text{is at the leftmost bit of the input} \end{array} \right\}.$$

Not applicable, but the language is in fact undecidable.

Rice's theorem cannot be used to prove the undecidability of this language!

Proof of Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Proof Idea:

Let P be a non-trivial property.

Assume that $\mathcal{L}_P$ is decidable.

Using this assumption prove that A_{TM} is decidable.

More specifically:

$$(M, w) \longrightarrow N$$

$$\begin{array}{lcl} \text{if } w \in L(M) & \longrightarrow & \langle N \rangle \in \mathcal{L}_P \\ \text{if } w \notin L(M) & \longrightarrow & \langle N \rangle \notin \mathcal{L}_P \end{array}$$

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Design of N

Let M_1 be the TM s.t. $L(M_1)$ has Property P.

Let $L(M_2)$ be the TM s.t. $L(M_2) = \varnothing$.

we assume that $\varnothing$ does not have property P[1]

on input x
{
if M accepts w
then if M_1 accepts x
then accept
}

Claim: $w \in L(M)$ if and only if $\langle N \rangle \in \mathcal{L}_P$

[1] We will remove this assumption later.

Getting rid of the assumption on P

We now show how to get around the assumption.

Suppose $\varnothing$ has property P.

Consider $\overline{P}$.

Now $\varnothing$ does not have property $\overline{P}$.

Use Rice's theorem on $\mathcal{L}_{\overline{P}}$ to prove undecidibility.

Conclude undecidibility of $\mathcal{L}_P$.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{ M \mid M \text{ has a useless state } \}$.

Not applicable, but the language is in fact undecidable.

Rice's theorem cannot be used to prove the undecidability of this language!

Summary of Module III

Introduction to Turing machines

Equivalent models

- multi-tape TM,
- non-deterministic TM

Turing decidable languages

Turing recognizable languages

Diagonalization in automata theory

Proving undecidability

- $A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$,
- $\text{Halt} = \{(M, w) \mid M \text{ hants on } w\}$,
- $E_{TM} = \{\langle M\rangle \mid L(M) = \varnothing\}$, $EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}$,
- $REG_{TM} = \{\langle M\rangle \mid L(M) \text{ is regular}\}$,

Rice's theorem

MPCP problem (Tutorial 11)

Notion of reduction (Tutorial 11)

At the end of last class

Undecidability of the following languages:

$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$.

$\text{Halt} = \{(M, w) \mid M \text{ hants on } w\}$.

$E_{TM} = \{\langle M \rangle \mid L(M) = \varnothing\}$.

$EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}$.

$REG_{TM} = \{\langle M \rangle \mid L(M) \text{ is regular}\}$.

Note that undecidability of REG_{TM} and E_{TM} can be proved using Rice's theorem.