Theory of Computation

Lesson 19a - Encodings

Review

Last Time:

1. All known models of Computation are equivalent to the TM.
   - Shown for the multitape machine
   - Discussed Java, ...
2. Church's Thesis:
   - Each algorithm can be implemented on a Turing machine
3. Hilbert's 10th problem:
   - recognizable, not decidable
4. Closure Properties
   - Closure properties for decidable: $A \cup B$, $A \cap B$, $\overline{A}$
   - Closure properties for recognizable: $A \cup B$, $A \cap B$

Review

Last Time:

5. A decidable $\Leftrightarrow$ A and $\overline{A}$ are recognizable
   - Hilbert undecidable $\Rightarrow$ Hilbert or $\overline{\text{Hilbert}}$ not recognizable
   - Hilbert recognizable $\Rightarrow$ $\overline{\text{Hilbert}}$ not recognizable
   - $\Rightarrow$ Recognizable languages not closed under complement
6. Enumerators (machines with attached printer)
   - A recognizable $\Leftrightarrow$ A has an enumerator (that prints A)
   - Proof with "clocks".

Encodings

Poll 19.1 (Graded on attendance)

Have you ever seen a natural number?

(a) Yes.

(b) No. $\longleftarrow$ correct

(c) Don't know.

(d) This question doesn't make sense.

Encodings

Encoding: Representation of objects as strings over the finite input alphabet Σ of a Turing machine or computer.

Object $\mathcal{O}$ $\longrightarrow$ encoding $<\mathcal{O}>$

Requirements:

(1) Must be able to (easily) decide whether a string is encoding $\{ w \in \Sigma^* \mid w = <\mathcal{O}> \text{ for some } \mathcal{O}\}$ must be (easily) decidable.

(2) Must be able to easily retrieve (all properties of) $\mathcal{O}$ from $<\mathcal{O}>$

Encodings

Example. Encoding in $\Sigma = \{0,1\}$

Objects: natural numbers

Encoding: $\langle n \rangle$ = bin(n) (binary representation of n)

Satisfy conditions?

(1) Given string $s \in \{0,1\}^*$, decide if s encodes some number?

Easy: (i) $s \neq \varepsilon$, (ii) no leading 0

(2) If $s = \langle n \rangle$ can easily recover properties of n? Yes

Encodings

Example. Polynomials with integer coefficients and variables $x_0, x_1, x_2, \ldots$

$p = 2x_1^3x_4^5 - 3x_2x_3^{10} + 4x_1^4x_3^2 + 5x_2 - 38$

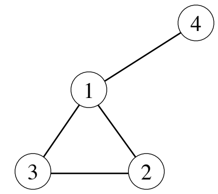
<p> = 2x_1^3x_4^5 - 3x_2x_3^{10} + 4x_1^4 x_3^2 + 5x_2 - 38

format: LaTeX

uses alphabet +, -, _, ^ , 0, 1, ..., 9, x , {, }

Encodings

Example. Graphs



Encoding 1:

$\underbrace{(1, 2, 3, 4)}_{\text{vertices}} \quad \underbrace{(\,(1, 2)\,, (2, 3)\,, (3, 1)\,, (1, 4)\,)}_{\text{edges}}$

Encoding 2:

	1	2	3	4
1	0	1	1	1
2	1	0	1	0
3	1	1	0	0
4	1	0	0	0

0111 | 1010 | 1100 | 1000

Encodings

Encoding of tuples.

tuple $(\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_k) \longrightarrow$ encoding $<\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_k>$

For example

$<\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_k> \; = \; < <\mathcal{O}_1> \# <\mathcal{O}_2> \# \ldots \# <\mathcal{O}_k> >$

where # is symbol not in the encoding alphabet of $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_k$

Requirement:

(3) Must be able to easily retrieve $<\mathcal{O}_1>, <\mathcal{O}_2>, \ldots, <\mathcal{O}_k>$ from $<\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_k>$

Encodings

Poll 19.2 (Graded on attendance)

Given sequence of binary numbers
say 10, 0, 11, 1010
How to encode over $\Sigma = \{0,1\}$?

(a) 100111010 ⟵ bad (can't retrieve the numbers)

(b) 110010001011111011001100 ⟵ good

fixed length encoding
0 ↔ 00
1 ↔ 11
, ↔ 10
$\langle \langle \mathcal{O}_1\rangle \# \langle \mathcal{O}_2\rangle \# \ldots \# \langle \mathcal{O}_k\rangle \rangle$

(c) neither

Encodings

For set S of objects, define language $\langle S\rangle = \{\langle \mathcal{O}\rangle \mid \mathcal{O} \in S\}$

Example. Even = $\{n \in \mathbb{N} \mid n \text{ is even}\}$

$\langle$Even$\rangle$ = binary strings without leading 0's that end with 0

Odd = $\{n \in \mathbb{N} \mid n \text{ is odd}\}$

$\langle$Odd$\rangle$ = binary strings without leading 0's that end with 1

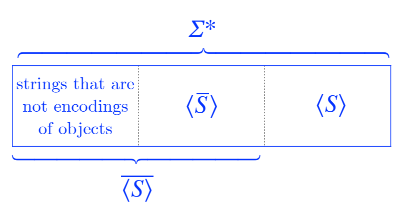
Question. Is $\langle \text{Odd}\rangle = \overline{\langle \text{Even}\rangle}$? $\overline{\langle \text{Even}\rangle} \neq \langle \overline{\text{Even}}\rangle$

No! $\overline{\langle \text{Even}\rangle}$ includes ε, 000, 010, ...

Encodings

For set S of objects, define language $\langle S\rangle = \{\langle \mathcal{O}\rangle \mid \mathcal{O} \in S\}$

Relationship between $\langle \overline{S}\rangle$ and $\overline{\langle S\rangle}$?



$\overline{\langle S\rangle} = \langle \overline{S}\rangle \cup \{\text{strings that are not encodings}\}$

thus

$\langle \overline{S}\rangle$ decidable/recognizable

$\Rightarrow \overline{\langle S\rangle}$ decidable/recognizable

$\langle \overline{S}\rangle = \overline{\langle S\rangle} \cap \{\text{strings that are encodings}\}$

thus

$\overline{\langle S\rangle}$ decidable/recognizable

$\Rightarrow \langle \overline{S}\rangle$ decidable/recognizable

That is,

$\langle \overline{S}\rangle$ decidable/recognizable $\Leftrightarrow$ $\overline{\langle S\rangle}$ decidable/recognizable

Dictionary for Informal People

Given set S of objects have defined $\langle S\rangle = \{ \langle \mathcal{O}\rangle \mid \mathcal{O} \in S\}$

informally say	exactly mean
S is decidable	$\langle S\rangle$ is decidable
S is recognizable	$\langle S\rangle$ is recognizable

Then

S decidable $\Leftrightarrow$ $\langle S\rangle$ decidable

$\Leftrightarrow$ $\langle S\rangle$ recognizable and $\overline{\langle S\rangle}$ recognizable

$\Leftrightarrow$ $\langle S\rangle$ recognizable and $\langle \overline{S}\rangle$ recognizable

$\Leftrightarrow$ S recognizable and $\overline{S}$ recognizable

Theory of Computation

Lesson 19b - Encoding Turing Machines

Encoding of Turing Machines

Goal: Want to feed (encodings of) Turing machines as input to other Turing machines.

Extreme case: Want to feed < T > as input to T !

Real life example (N. Wirth): PASCAL compilers are PASCAL programs. A PASCAL compiler must be able to compile itself (very quickly).

Encoding of Turing Machines

Requirement: Turing machines with input alphabet Σ should be encoded in Σ.

Example. Encoding of $M = (Q, \Sigma, \Gamma, \delta, s, q_{accept}, q_{reject})$ where $\Sigma = \{0, 1\}$

(1) Assign a number to each symbol in Γ and state in Q

$\Gamma = \{a_0, a_1, a_2, a_3, \dots\}$

whereby $a_0 = 0$, $a_1 = 1$, $a_2 = \sqcup$

$Q = \{q_0, q_1, q_2, q_3, \dots\}$

whereby $q_0 = s$, $q_1 = q_{accept}$, $q_2 = q_{reject}$

Encoding of Turing Machines

(2) To specify M only need to know the quintuples

- code(q_i, a_j, q_k, a_ℓ, move) = q bin(i) a bin(j) q bin(k) a bin(ℓ) move

  example: code(q_3, a_5, q_2, a_7, R) = $q11a101q10a111R$

- If quintuples are $\delta_1, \delta_2, \dots, \delta_n$ then

  code(M) = code(δ_1) code(δ_2) ... code(δ_n)

Yields encoding of M over the alphabet $\{q, a, L, R, S, 0, 1\}$

Encoding of Turing Machines

(3) Encode M over the alphabet $\Sigma = \{0,1\}$

Replace symbols $q, a, L, R, S, 0, 1$ with 3-bit strings

0	1	q	a	L	R	S	#
000	111	011	110	100	001	010	101

Notice that # is not used

Encoding of Turing Machines

Exercise. Let $\Sigma = \{0,1\}$. If Turing machines are encoded as described on the last three slides, what is the language accepted by the machine M with encoding below?

$< M > \ =$

011 000 110 000 011 000 110 111 010 011 000 110 111 011 111 110 000 100 011 000 110 111 000 011 111 000 110 111 010

CQ - Encoding of Turing Machines

Question. If $\Sigma = \{1\}$ (has only one symbol) how do we encode Turing machines from alphabet $\{q, a, L, R, S, 0, 1\}$ into Σ ?

Answer. Turing machine T is string over $\{q, a, L, R, S, 0, 1\}$.

Encode as 1^n where n = shortlex number of T

Encoding Turing Machines

Theorem. There exists a universal Turing machine $\mathcal{U}$:
On input $< M, w >$ where
(1) M is a Turing machine with input alphabet Σ and
(2) $w \in \Sigma^*$,
$\mathcal{U}$ simulates M started on w.
($\mathcal{U}$ accepts/rejects/loops if M accepts/rejects/loops.)

Theorem discovered before stored program machines were invented

Proof. Construct 3-tape machine T that on input $<M, w>$ on tape 1,
- writes w on tape 2
- writes $<M>$ on tape 3
- runs M (on input w) on tape 2.

$\mathcal{U}$: one-tape machine simulating T.

Theory of Computation

Lesson 19c - Encoding Computations + application to Enumerable Languages

Encoding Computations

Computation history of T on input w is a sequence of configurations

$$C_0, C_1, \dots, C_n$$

such that
1. C_0 is the start configuration of T on input w
2. Each configuration C_i with $i > 0$ results from the previous configuration by application of a transition of T.

Computation history is accepting / rejecting if last configuration is in state q_{accept} / q_{reject} .

Encoding Computations

Computation histories can be encoded

$$C_0, C_1, \dots, C_n \longrightarrow < C_0, C_1, \dots, C_n >$$

- As before, can encode a computation first in $\{0, 1, q, a, L, R, S, \#\}$ then in $\{0,1\}$.
- Given machine M and string s, it is easy to check whether s encodes a computation history of M.

Encodings

Poll 19.3 In lecture 18 we proved

L is recognizable $\Leftrightarrow$ L has an enumerator

Which direction was more difficult, i.e. needed clocks?

(a) $\Rightarrow$ ⟵ needed clocks

(b) $\Leftarrow$

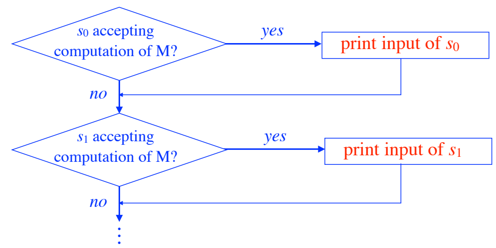
Enumerators

Theorem 4 (revisited).

Language L is recognizable $\Leftrightarrow$ there exists an enumerator for L.

Proof of $\Rightarrow$: Suppose L is recognized by machine M.

Let $s_0, s_1, s_2, \ldots$ be the strings of Σ^* in shortlex order



Enumerators by Computation Encoding

Theorem (revisited). Let $L \subseteq \Sigma^*$. Then L recognizable $\Leftrightarrow$ L has an enumerator.

Proof (more formal).

$\Leftarrow$: As before. Suppose that L has an enumerator. Recognizer takes input w on tape 1, then runs enumerator on tapes 2, 3. Each word printed is compared with w. If match, accept w.

$\Rightarrow$: Suppose that L is recognized by machine M and let $s_0, s_1, s_2, \ldots$ be the words of Σ^* in shortlex order.

for $i = 0, 1, 2, \ldots$ **do**

if s_i is the encoding of an accepting computation of M **then** print the input of the computation.

uses iterator for Σ^*

Theory of Computation

Lesson 19d - Acceptance Problems

The Acceptance Problem

Assume fixed $\Sigma = \{0,1\}$ throughout this discussion.

Acceptance Problem:

DFA, NFA, REX, CFG, PDA, TM

Given: • Computation Model C with alphabet Σ

• String $w \in \Sigma^*$

Decide: whether $w \in L(C)$

How is (C, w) "given" to computer or TM?

☞ As encoded pair $\langle C, w \rangle$

Acceptance Problem for DFAs

Problem reformulated as language:

Definition. Acceptance Problem for DFAs.

$A_{DFA} = \{\langle M,w \rangle \mid M \text{ is a DFA and } M \text{ accepts } w\}$

more formally:

$A_{DFA} = \{\langle M,w \rangle \mid M$ is a DFA with input alphabet Σ, $w \in \Sigma^*$, and M accepts $w\}$

Acceptance Problem for DFAs

Acceptance Problem for DFAs.

$A_{DFA} = \{\langle M,w\rangle \mid M$ is a DFA and M accepts $w\}$

Theorem 1. A_{DFA} is decidable

Proof: Give algorithm (Turing machine) to decide $u \in A_{DFA}$:

On input string $u \in \Sigma^*$

1. Reject if u is not encoding of the form $\langle M, w\rangle$ with M a DFA
2. Otherwise simulate M on input w
   i) accept u if M accepts w
   ii) reject u if M doesn't accept w.

Acceptance Problem for NFAs

Definition. Acceptance Problem for NFAs.

$A_{NFA} = \{\langle M,w\rangle \mid M$ is a NFA and M accepts $w\}$

Theorem 2. A_{NFA} is decidable

Proof: Give algorithm to decide $u \in A_{NFA}$:

On input string $u \in \Sigma^*$

1. Reject if u is not encoding of the form $\langle M, w\rangle$
2. Otherwise convert M to DFA $\hat{M}$
   - accept u if $\hat{M}$ accepts w
   - reject u if $\hat{M}$ doesn't accept w.

Acceptance Problem for NFAs

Definition. Acceptance Problem for NFAs.

$A_{NFA} = \{\langle M,w\rangle \mid M$ is a NFA and M accepts $w\}$

Theorem 2. A_{NFA} is decidable

Proof by reduction: Give algorithm to decide $u \in A_{NFA}$:

On input string $u \in \Sigma^*$

1. Reject if u is not encoding of the form $\langle M, w\rangle$
2. Otherwise convert M to DFA $\hat{M}$
   Give $\langle \hat{M}, w\rangle$ as input to Turing machine T of Theorem 1.
   - accept u if T accepts $\langle \hat{M}, w\rangle$
   - reject u if T rejects $\langle \hat{M}, w\rangle$.