1

Outline

- Post Correspondence Problem (skip)
- Beyond RE and co-RE
- Recursion Theorem

- On to complexity…

2

Post Correspondence Problem

- many undecidable problems unrelated to TMs and automata

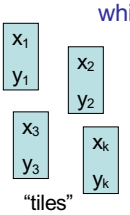
- classic example: Post Correspondence Problem

PCP = {<(x_1, y_1), (x_2, y_2), …, (x_k, y_k)> :
x_i, y_i ∈ Σ* and there exists (a_1, a_2, …, a_n) for which $x_{a_1}x_{a_2}\ldots x_{a_n} = y_{a_1}y_{a_2}\ldots y_{a_n}$}

3

Post Correspondence Problem

PCP = {<(x_1, y_1), (x_2, y_2), …, (x_k, y_k)> :
x_i, y_i ∈ Σ* and there exists (a_1, a_2, …, a_n) for which $x_{a_1}x_{a_2}\ldots x_{a_n} = y_{a_1}y_{a_2}\ldots y_{a_n}$}



"tiles"

x_2	x_1	x_5	x_2	x_1	x_3	x_4	x_4
y_2	y_1	y_5	y_2	y_1	y_3	y_4	y_4

$x_2x_1x_5x_2x_1x_3x_4x_4 = y_2y_1y_5y_2y_1y_3y_4y_4$

"match"

4

Post Correspondence Problem

Theorem: PCP is undecidable.

Proof:
- reduce from A_{TM} (i.e. show $A_{TM} \leq_m$ PCP)
- two step reduction makes it easier
- first, show $A_{TM} \leq_m$ MPCP
  (MPCP = "modified PCP")
- next, show MPCP $\leq_m$ PCP

5

Post Correspondence Problem

MPCP = {<(x_1, y_1), (x_2, y_2), …, (x_k, y_k)> :
x_i, y_i ∈ Σ* and there exists (a_1, a_2, …, a_n) for which $x_1x_{a_1}x_{a_2}\ldots x_{a_n} = y_1y_{a_1}y_{a_2}\ldots y_{a_n}$}

Proof of MPCP $\leq_m$ PCP:
- notation: for a string $u = u_1u_2u_3\ldots u_m$
  - $\star u$ means the string $\star u_1\star u_2\star u_3\star u_4\ldots\star u_m$
  - $u\star$ means the string $u_1\star u_2\star u_3\star u_4\ldots\star u_m\star$
  - $\star u\star$ means the string $\star u_1\star u_2\star u_3\star u_4\ldots\star u_m\star$

6

Post Correspondence Problem

Proof of MPCP $\leq_m$ PCP:
- given an instance $(x_1, y_1), \ldots, (x_k, y_k)$ of MPCP
- produce an instance of PCP:
  $(\star x_1, \star y_1 \star), (\star x_1, y_1 \star), (\star x_2, y_2 \star), \ldots, (\star x_k, y_k \star), (\star \square, \square)$
- YES maps to YES?
  - given a match in original MPCP instance, can produce a match in the new PCP instance
- NO maps to NO?
  - given a match in the new PCP instance, can produce a match in the original MPCP instance

7

Post Correspondence Problem

- YES maps to YES?
  - given a match in original MPCP instance, can produce a match in the new PCP instance

x_1	x_4	x_5	x_2	x_1	x_3	x_4	x_4
y_1	y_4	y_5	y_2	y_1	y_3	y_4	y_4

$\star x_1$	$\star x_4$	$\star x_5$	$\star x_2$	$\star x_1$	$\star x_3$	$\star x_4$	$\star x_4$	$\star \square$
$\star y_1 \star$	$Y_4 \star$	$y_5 \star$	$y_2 \star$	$y_1 \star$	$y_3 \star$	$y_4 \star$	$y_4 \star$	$\square$

8

Post Correspondence Problem

- NO maps to NO?
  - given a match in the new PCP instance, can produce a match in the original MPCP instance

can't match unless start with this tile

$\star x_1$	$\star x_4$	$\star x_5$	$\star x_2$	$\star x_1$	$\star x_3$	$\star x_4$	$\star x_4$	$\star \square$
$\star y_1 \star$	$Y_4 \star$	$y_5 \star$	$y_2 \star$	$y_1 \star$	$y_3 \star$	$y_4 \star$	$y_4 \star$	$\square$

x_1	x_4	x_5	x_2	x_1	x_3	x_4	x_4
y_1	y_4	y_5	y_2	y_1	y_3	y_4	y_4

$\star$ symbols must align

can only appear at the end

9

Post Correspondence Problem

Theorem: PCP is undecidable.

Proof:
- show $A_{TM} \leq_m$ MPCP

  MPCP = $\{\langle (x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k) \rangle :$
  $x_i, y_i \in \Sigma^*$ and there exists $(a_1, a_2, \ldots, a_n)$ for which $x_1 x_{a_1} x_{a_2} \ldots x_{a_n} = y_1 y_{a_1} y_{a_2} \ldots y_{a_n}\}$

- show MPCP $\leq_m$ PCP

10

Post Correspondence Problem

Proof of $A_{TM} \leq_m$ MPCP:
- given instance of A_{TM}: <M, w>
- idea: a match will record an accepting computation history for M on input w
- start tile records starting configuration:
  - add tile $(\#, \#q_0 w_1 w_2 w_3 \ldots w_n \#)$

#		#
$\#q_0 w_1 w_2 \ldots w_n \#$	=	$\#C_1 \#$

11

Post Correspondence Problem

#	?	?	…	?	=	$\#C_1 \#$
$\# q_0 w_1 w_2 \ldots w_n \#$	?	?		?		$\#C_1 \#C_2 \#$

- tiles for head motions to the right:
  - for all $a,b \in \Gamma$ and all $q, r \in Q$ with $q \neq q_{reject}$, if $\delta(q, a) = (r, b, R)$, add tile $\begin{matrix} qa \\ br \end{matrix}$
- tiles for head motions to the left:
  - for all $a,b,c \in \Gamma$ and all $q, r \in Q$ with $q \neq q_{reject}$, if $\delta(q, a) = (r, b, L)$, add tile $\begin{matrix} cqa \\ rcb \end{matrix}$

12

Post Correspondence Problem

#	?	?	…	?	=	#C_1#
# $q_0w_1w_2…w_n$#	?	?		?		#C_1#C_2#

– tiles for copying (not near head)
  • for all a ∈ Γ, add tile (a, a)

a
a

– tiles for copying # marker
  • add tile (#, #)

#
#

– tiles for copying # marker and adding _ to end of tape
  • add tile (#, _#)

#
_#

13

Post Correspondence Problem

#	?	…	?	=	#$uaq_{accept}v$#
#$uaq_{accept}v$#	?		?		#$uaq_{accept}v$#$uq_{accept}v$#

– tiles for deleting symbols to left of q_{accept}
  • for all a ∈ Γ, add tile (aq_{accept}, q_{accept})

aq_{accept}
q_{accept}

14

Post Correspondence Problem

#	?	…	?	=	#$q_{accept}av$#
#$q_{accept}av$#	?		?		#$q_{accept}av$#$q_{accept}v$#

– tiles for deleting symbols to right of q_{accept}
  • for all a ∈ Γ, add tile ($q_{accept}a$, q_{accept})

$q_{accept}a$
q_{accept}

15

Post Correspondence Problem

#	?	…	?	=	#q_{accept}##
#q_{accept}#	?		?		#q_{accept}##

– tiles for completing the match
  • for all a ∈ Γ, add tile (q_{accept}##, #)

q_{accept}##
#

16

Post Correspondence Problem

– YES maps to YES?
  • by construction, if M accepts w, there is a way to assemble the tiles to achieve this match:

#C_1#C_2#C_3#...#C_m#
#C_1#C_2#C_3#...#C_m#

where #C_1#C_2#C_3#...#C_m# is an accepting computation history

– NO maps to NO?
  • sketch: at any step if the "intended" next tile is not used, then it is impossible to recover and produce a match in the end (case analysis)

17

Post Correspondence Problem

We have proved:

Theorem: PCP is undecidable.

by showing:
– $A_{TM} \leq_m$ MPCP
– MPCP $\leq_m$ PCP
– conclude $A_{TM} \leq_m$ PCP

18

Beyond RE and co-RE

- We saw (by a counting argument) that there is *som* that is ... or co-RE.
- We will prove this for a natural language:
  $EQ_{TM} = \{\langle M_1, M_2\rangle : L(M_1) = L(M_2)\}$
- Recall:
  - A_{TM} is undecidable, but RE
  - co-A_{TM} is undecidable, but coRE

Therefore, not in co-RE

Therefore, not in RE

19

Beyond RE and co-RE

Theorem: EQ_{TM} is neither RE nor coRE.

Proof:

- not RE:
  - reduce from co-A_{TM} (i.e. show co-$A_{TM} \leq_m EQ_{TM}$)
  - what should f(<M, w>) produce?
- not co-RE:
  - reduce from A_{TM} (i.e. show $A_{TM} \leq_m EQ_{TM}$)
  - what should f(<M, w>) produce?

20

Beyond RE and co-RE

Proof ($A_{TM} \leq_m EQ_{TM}$)

- f(<M, w>) = <M_1, M_2> described below:

TM M_1: on input x,
- accept

TM M_2: on input x,
- simulate M on input w
- accept if M accepts w

- YES maps to YES?
  $\langle M, w\rangle \in A_{TM} \Rightarrow L(M_1) = \Sigma^*$ and $L(M_2) = \Sigma^*$
  $\Rightarrow f(\langle M, w\rangle) \in EQ_{TM}$
- NO maps to NO?
  $\langle M, w\rangle \notin A_{TM} \Rightarrow L(M_1) = \Sigma^*$ and $L(M_2) = \emptyset$
  $\Rightarrow f(\langle M, w\rangle) \notin EQ_{TM}$

21

Beyond RE and co-RE

Proof (co-$A_{TM} \leq_m EQ_{TM}$)

- f(<M, w>) = <M_1, M_2> described below:

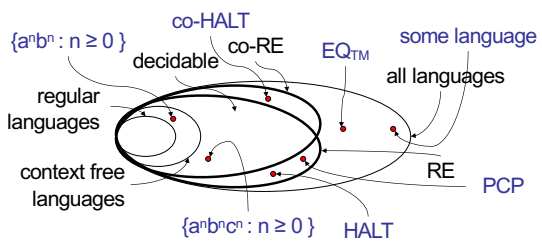
TM M_1: on input x,
- reject

TM M_2: on input x,
- simulate M on input w
- accept if M accepts w

- YES maps to YES?
  $\langle M, w\rangle \in$ co-A_{TM}
  $\Rightarrow L(M_1) = \emptyset$ and $L(M_2) = \emptyset$
  $\Rightarrow f(\langle M, w\rangle) \in EQ_{TM}$
- NO maps to NO?
  $\langle M, w\rangle \notin$ co-A_{TM}
  $\Rightarrow L(M_1) = \emptyset$ and $L(M_2) = \Sigma^*$
  $\Rightarrow f(\langle M, w\rangle) \notin EQ_{TM}$

22

Summary



23

The Recursion Theorem

- A very useful, and non-obvious, capability of Turing Machines:
  - in the course of computation, can print out a description of itself!
- how is this possible?
  - an example of a program that prints out self:

Print two copies of the following, the 2nd one in quotes:
"Print two copies of the following, the 2nd one in quotes:"

24

The Recursion Theorem

- Why is this useful?
- Example: slick proof that A_{TM} undecidable
  - assume TM M decides A_{TM}
  - construct machine M' as follows:

on input x,
- obtain own description <M'>
- run M on input <M', x>
- if M rejects, accept; if M accepts, reject.

if M' on input x:
- accepts, then M rejects <M', x>, but then M' does not accept!
- rejects, then M accepts <M', x>, but then M' accepts!

25

The Recursion Theorem

- Lemma: there is a computable function
  $q:\Sigma^* \rightarrow \Sigma^*$
  such that q(w) is a description of a TM P_w that prints out w and then halts.
- Proof:
  - on input w, construct TM P_w that has w hard-coded into it; output <P_w>

26

The Recursion Theorem

- Warm-up: produce a TM SELF that prints out its own description.
- Two parts:
  - Part A:
    - output a description of B
    - pass control to B.
  - Part B:
    - prepend a description of A
    - done

27

The Recursion Theorem

- Part A:
  - output a description of B
  - pass control to B.
- Part B:
  - prepend a description of A
  - done

Recall: q(w) is a description of a TM P_w that prints out w and then halts.

Note: <A> = q(<B>)

A
- output <B>

B
- read contents of tape
- apply q to it
- prepend* result to tape

*combine with description on tape to produce a complete TM

28

The Recursion Theorem

Note: <A> = q(<B>)

A
- output <B>

B
- read contents of tape
- apply q to it
- prepend result to tape

Recall: q(w) is a description of a TM P_w that prints out w and then halts.

- watch closely as TM AB runs:
- A runs. Tape contents: <B>
- B runs. Tape contents: q(<B>)<B> = <AB>
- AB is our desired machine SELF.

29

The Recursion Theorem

- Lemma: there is a computable function
  $q:\Sigma^* \rightarrow \Sigma^*$
  such that q(w) is a description of a TM P_w that prints out w and then halts.
- Proof:
  - on input w, construct TM P_w that has w hard-coded into it; output <P_w>

30

The Recursion Theorem

- Warm-up: produce a TM SELF that prints out its own description.
- Two parts:
  - Part A:
    - output a description of B
    - pass control to B.
  - Part B:
    - prepend a description of A
    - done

31

The Recursion Theorem

- Part A:
  - output a description of B
  - pass control to B.
- Part B:
  - prepend a description of A
  - done

Recall: q(w) is a description of a TM P_w that prints out w and then halts.

Note: <A> = q(<B>)

A
• output <B>

B
• read contents of tape
• apply q to it
• prepend* result to tape

*combine with description on tape to produce a complete TM

32

The Recursion Theorem

Note: <A> = q(<B>)

A
• output <B>

Recall: q(w) is a description of a TM P_w that prints out w and then halts.

B
• read contents of tape
• apply q to it
• prepend result to tape

- watch closely as TM AB runs:
- A runs. Tape contents: <B>
- B runs. Tape contents: q(<B>)<B> = <AB>
- AB is our desired machine SELF.

33

The Recursion Theorem

Theorem: Let T be a TM that computes fn:

$t: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$

There is a TM R that computes the fn:

$r: \Sigma^* \rightarrow \Sigma^*$

defined as $r(w) = t(w, \langle R \rangle)$.

- This allows "obtain own description" as valid step in TM program
  - first modify TM so that it takes an additional input (that is own description); use at will

34

The Recursion Theorem

Theorem: Let T be a TM that computes fn:

$t: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$

There is a TM R that computes the fn:

$r: \Sigma^* \rightarrow \Sigma^*$

defined as $r(w) = t(w, \langle R \rangle)$.

Proof outline: TM R has 3 parts

Part A: output description of BT

Part B: prepend description of A

Part "T": run TM T

35

The Recursion Theorem

Proof details: TM R has 3 parts

Part A: output description of BT

- <A> = q(<BT>)

Part B: prepend description of A

- read contents of tape <BT>
- apply q to it q(<BT>) = <A>
- prepend to tape <ABT>

Part "T": run TM T

- 2nd argument on tape is description of R

36

Summary

- full-fledged model of computation: TM
- many equivalent models
- Church-Turing Thesis

- encoding of inputs
- Universal TM

37

Summary

- classes of problems:
  - decidable ("solvable by algorithms")
  - recursively enumerable (RE)
  - co-RE

- counting:
  - not all problems are decidable
  - not all problems are RE

38

Summary

- diagonalization: HALT is undecidable
- reductions: other problems undecidable
  - many examples
  - Rice's Theorem
- natural problems that are not RE
- Recursion Theorem: non-obvious capability of TMs: printing out own description
- Incompleteness Theorem

39

Complexity

- So far we have classified problems by whether they have an algorithm at all.
- In real world, we have limited resources with which to run an algorithm:
  - one resource: time
  - another: storage space
- need to further classify decidable problems according to resources they require

40

Complexity

- Complexity Theory = study of what is computationally feasible (or tractable) with limited resources:
  - running *time* — main focus
  - storage *space*
  - number of *random bits*
  - degree of *parallelism*
  - rounds of *interaction*
  - *others…*

  } not in this course

41

Worst-case analysis

- Always measure resource (e.g. running time) in the following way:
  - as a function of the input length
  - value of the fn. is the maximum quantity of resource used over all inputs of given length
  - called "worst-case analysis"
- "input length" is the length of input string, which might encode another object with a separate notion of size

42