CS5371
Theory of Computation

Lecture 15: Computability VI
(Post's Problem, Reducibility)

Objectives

- In this lecture, we introduce Post's correspondence problem, which gives an undecidable language which is based on dominos
- We also introduce computable functions, which allows us to look at reduction in a formal way

Post's Correspondence Problem

Let P be a finite set of dominoes $\{d_1, d_2, \ldots, d_k\}$, each piece of domino d_i consists of a top string t_i and a bottom string b_i

A match in P is a sequence i1, i2,..., ij (allowing repeats) such that $t_{i1}\ t_{i2} \ldots t_{ij} = b_{i1}\ b_{i2} \ldots b_{ij}$. That is, we can find a sequence of dominoes such that the concatenation of the top strings is equal to the concatenation of the bottom strings

Post's Correspondence (2)

Let PCP be the language

$\{\langle P \rangle \mid P \text{ is a set of dominoes with a match}\}$

Theorem: PCP is undecidable

Post's Correspondence (3)

Before we do that, let us modified the problem a bit… We require a match is this problem must start with the first domino

Let MPCP be the language

{⟨P⟩ | P is a set of dominoes with a match starting with the first domino}

Theorem: MPCP is undecidable

Proving MPCP

Proof Idea:

Prove by reducing A_{TM} to MPCP. That is, assuming MPCP is decidable, we then show A_{TM} is decidable.

On input M, w, let us design a set of dominoes, such that whenever M accepts w, there is a match, and whenever M does not accept w, there is no match

Proving MPCP (2)

Difficulty: We need to correspond every possible (I.e., infinitely many) computation of TM just by using the dominos …

Observation: From a configuration to the next configuration, the change is "local" -- only around the tape head, and number of possible changes are finite

On input M, w, we design dominos such that M accepts w if and only if the match corresponds to an accepting computation history for M to accept w

Proving MPCP (3)

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{reject})$ and $w = w_1 w_2 \ldots w_n$

Step 1: Put $\# \mid \# q_0 w_1 w_2 \ldots w_n \#$ as the first domino (the notation t|b denote t is the top string, and b is the bottom string)

Step 2: For every a,b in Γ, every q,r in Q where $q \neq q_{reject}$

Create domino qa | br if $\delta(q,a) = (r,b,R)$

Proving MPCP (4)

Step 3: For every a,b,c in Γ, every q,r in Q where $q \neq q_{reject}$

Create domino cqa | rcb if $\delta(q,a) = (r,b,L)$

How to handle the boundary case (when the tape head is at the leftmost end of tape?)

Step 4: For every a in Γ,

Create domino a | a

Proving MPCP (5)

Step 5: Create dominoes # | # and # | □#

Why do we need # | □# ?

Step 6: For every a in Γ

Create dominoes aq_{accept} | q_{accept} and $q_{accept}a$ | q_{accept}

Step 7: For every a in Γ

Create dominoes q_{accept}## | #

Proving MPCP (6)

We claim that MPCP has a match if and only if M accepts w. Thus, MPCP is undecidable (why? Can you construct a decider for A_{TM} based on a decider of MPCP?)

It remains to show PCP is undecidable … We prove this by reducing MPCP to PCP

Reducing MPCP to PCP

We use a trick to transform any MPCP problem to a PCP problem, while enforcing the match must start with the first domino

Before that, we introduce the following notation: for any string $u = u_1 u_2 \dots u_k$

$*u* = * u_1 * u_2 * \dots * u_k *$

$u* = \quad u_1 * u_2 * \dots * u_k *$

$*u \quad = * u_1 * u_2 * \dots * u_k$

Reducing MPCP to PCP (2)

Let $P' = \{d_1, d_2, \ldots, d_k\}$ be the dominos in MPCP, with d_i consisting a top string t_i and bottom string b_i; d_1 is the first domino

We construct P as follows:

1. Add $*t_1 \mid *b_1*$ to P
2. Add $*t_j \mid b_j*$ to P, for every $j = 1,2,\ldots,k$
3. Add $*\blacklozenge \mid \blacklozenge$ to P

Thus, P has a match if and only if P' has a match starting with d_1

What is a Reduction?

We have seen a lot of examples that applies reduction technique

Here, we give one formal way of defining reduction → This allows us to understand more about the power of reduction, and allows us to prove more results

The formal definition is based on computable functions (see next slides)

Computable Function

A Turing machine can compute a function as follows: at the beginning, the tape contains the input; once it halts, the tape contains the output

A function $f: \Sigma^* \rightarrow \Sigma^*$ is a computable function if some TM exists such that for any input w, it halts with $f(w)$ on its tape

E.g., all arithmetic operations on integers are computable functions (try prove it at home)

Defining Reduction

Definition: Language A is mapping reducible to language B, written as $A \leq_m B$, if there is a computable function $f: \Sigma^* \rightarrow \Sigma^*$ such that for every w,

$$w \in A \Leftrightarrow f(w) \in B$$

The function f is called the reduction of A to B

Some results

Theorem 1: If $A \leq_m B$ and B is decidable, then A is decidable. (Why?)

Theorem 2: If $A \leq_m B$ and A is undecidable, then B is undecidable.

Some results (2)

Theorem 3: If $A \leq_m B$ and B is recognizable, then A is recognizable (Why?)

Theorem 4: If $A \leq_m B$ and A is non-recognizable, then B is non-recognizable

Examples ($HALT_{TM}$)

A long time ago, we showed how to reduce A_{TM} to $HALT_{TM}$

To show this by mapping reduction, we want to find a computable function f that performs as follows:

If $x = \langle M, w \rangle$, $f(x) = \langle M', w' \rangle$

such that $x \in A_{TM} \Leftrightarrow f(x) \in HALT_{TM}$

Else, $f(x) = \varepsilon$ (or, pick any other string not in $HALT_{TM}$)

Examples ($HALT_{TM}$)

Then, f can be computed by the TM F below:

F = "On input $\langle M, w \rangle$,

1. Construct the machine M':

   M' = "On input x

   1. Run M on x

   2. If M accepts, accept; Else, enter loop"

2. Output $\langle M', w \rangle$

Thus, f is a computable function, so that

$A_{TM} \leq_m HALT_{TM}$

Examples (PCP)

In PCP problem, we showed how to reduce A_{TM} to MPCP

To show this by mapping reduction, we want to find a computable function f that:

If $x = \langle M, w \rangle$, $f(x) = \langle P \rangle$

such that $x \in A_{TM} \Leftrightarrow f(x) \in MPCP$

Else, $f(x) = \varepsilon$ (or any $\langle P \rangle$ not in MPCP)

We can construct a TM that computes f (how?). Thus, $A_{TM} \leq_m MPCP$

Examples (PCP)

Also, we showed how to reduce MPCP to PCP

To show this by mapping reduction, we want to find a computable function g that:

If $x = \langle P \rangle$, $g(x) = \langle P' \rangle$

such that $x \in MPCP \Leftrightarrow g(x) \in PCP$

Else, $g(x) = \varepsilon$

We can construct a TM that computes g, so that $PCP \leq_m PCP$

Examples (PCP)

Combining the two examples, we can argue that the function $h = g \circ f$ is also a computable function (why?), and has the property that:

$$x \in A_{TM} \Leftrightarrow h(x) \in PCP \text{ (why?)}$$

Thus, $A_{TM} \leq_m PCP$. By Theorem 1, we conclude that PCP is undecidable

Examples (E_{TM})

When we show E_{TM} is undecidable, our proof is by reducing A_{TM} to E_{TM}

Let us recall a bit how we do so:

On given any input $\langle M, w \rangle$, we construct a TM M' such that

if M accepts w, then $L(M') = \{w\}$

if M not accept w, then $L(M') = \{\ \}$

This in fact gives us a computable function f reducing A_{TM} to "complement of E_{TM}"

Examples (E_{TM})

I.e., we have a computable function f that:

$$x \in A_{TM} \Leftrightarrow f(x) \notin E_{TM},$$

or equivalently,

$$x \in A_{TM} \Leftrightarrow f(x) \in E'_{TM}$$

where E'_{TM} denotes the complement of E_{TM}

Thus, $A_{TM} \leq_m E'_{TM}$, which still implies E_{TM} is undecidable (why?)

Question: Can we prove the above by finding a mapping reduction of A_{TM} to E_{TM}?

Examples (E_{TM})

... the answer is NO (Exercise 5.5)

Proof: Suppose on the contrary that $A_{TM} \leq_m E_{TM}$. Then, we have $A'_{TM} \leq_m E'_{TM}$ (why?).

However, E'_{TM} is recognizable (why?) but A'_{TM} is not recognizable...

Thus, contradiction occurs (where?) and we conclude that no reduction of A_{TM} to E_{TM} exists

Examples (EQ_{TM})

We have seen one example of a non-Turing recognizable language: A'_{TM}

Define: A language is co-recognizable if its complement is recognizable.

Then, we have:

Theorem: EQ_{TM} is not recognizable, and not co-recognizable. That is, EQ_{TM} is not recognizable, and EQ'_{TM} is not recognizable.

Examples (EQ_{TM})

Proof: We first show that $A_{TM} \leq_m EQ'_{TM}$.

If this can be shown, we equivalently has shown that $A'_{TM} \leq_m EQ_{TM}$ (why?) and proved that EQ_{TM} is not recognizable.

To show $A_{TM} \leq_m EQ'_{TM}$, we construct the TM

F giving the desired reduction f as follows (see next slide):

Examples (EQ_{TM})

F = "On input ⟨M, w⟩,

1. Construct machines M1 and M2:

   M1 = "On any input,

   1. Reject"

   M2 = "On any input,

   1. Run M on w. If it accepts, accept"

2. Output ⟨M1, M2⟩"

So, on input x = ⟨M, w⟩, F computes ⟨M1, M2⟩ as f(x). What is the property of f(x)?

Examples (EQ_{TM})

Next, we show that $A_{TM} \leq_m EQ_{TM}$. If this can be shown, we equivalently has shown that $A'_{TM} \leq_m EQ'_{TM}$ and proved that EQ'_{TM} is not recognizable.

To show $A_{TM} \leq_m EQ_{TM}$, we construct the TM G giving the desired reduction g as follows (see next slide):

Examples (EQ_{TM})

G = "On input ⟨M, w⟩,

1. Construct machines M1 and M2:

   M1 = "On any input,

   1. Accept"

   M2 = "On any input,

   1. Run M on w. If it accepts, accept"

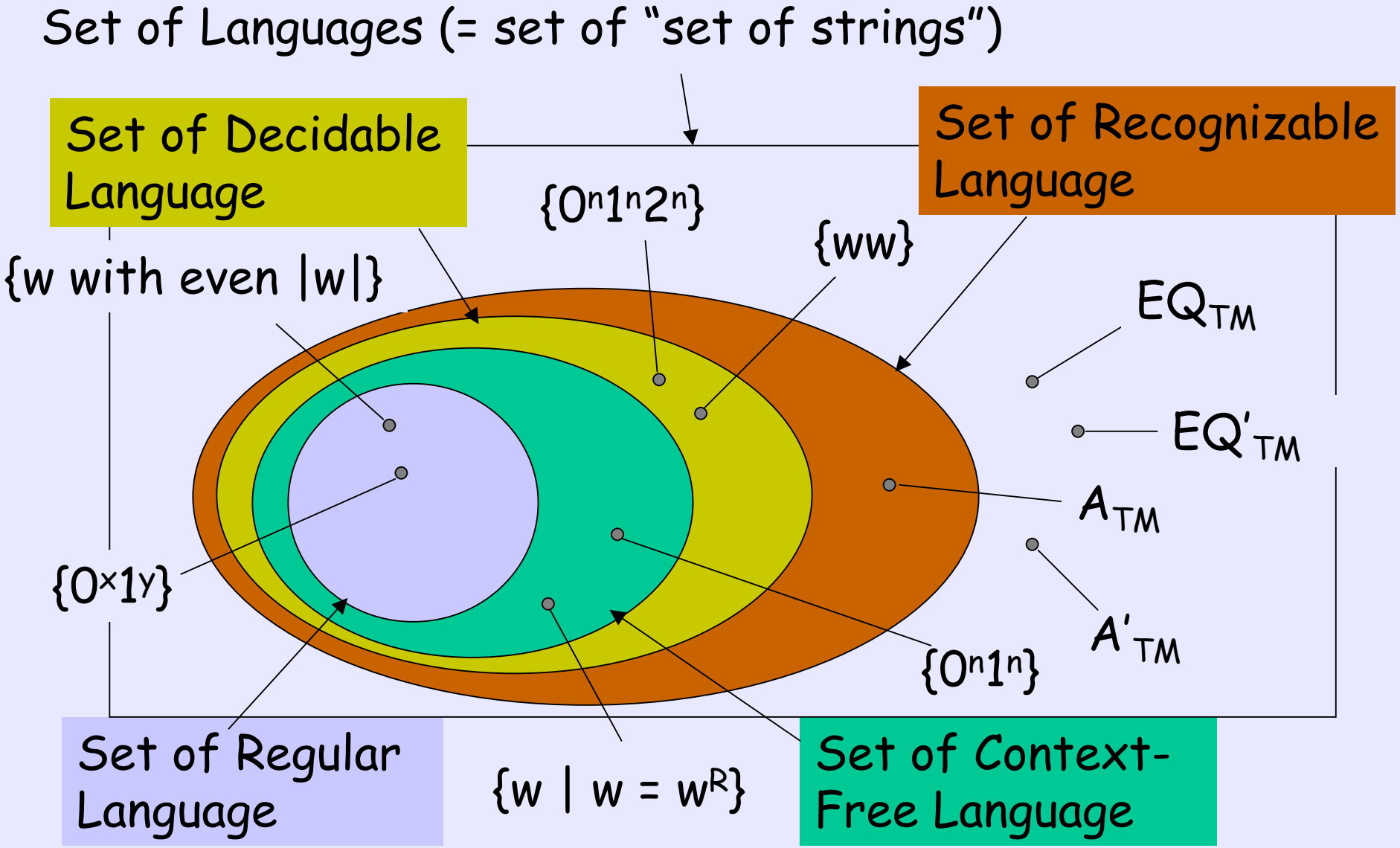
2. Output ⟨M1, M2⟩"

So, on input x = ⟨M, w⟩, G computes ⟨M1, M2⟩ as g(x). What is the property of g(x)?

What we have learnt

- $HALT_{TM}$, E_{TM}, $REGULAR_{TM}$, EQ_{TM}, PCP are undecidable (reduction from A_{TM})
- A_{LBA} is decidable (finite test cases)
- E_{LBA} and ALL_{CFG} are undecidable (reduction from A_{TM} via computation history)
- Computable function, mapping reducibility
- EQ_{TM} and EQ'_{TM} are non-recognizable, (reduction from A'_{TM})

Language Hierarchy (revisited)



Next Time

- Complexity Theory
  - To classify the problems based on the resources (time or memory usage)