Lec 20. Reduction and undecidable languages III

Eunjung Kim

Post Correspondence Problem (PCP)

$$\left\{ \left[\frac{\mathtt{b}}{\mathtt{ca}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right], \left[\frac{\mathtt{ca}}{\mathtt{a}}\right], \left[\frac{\mathtt{abc}}{\mathtt{c}}\right] \right\}$$ Chapter 5.2, Sipser 2012.

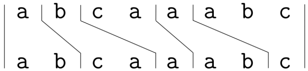
Emil Post's Correspondence Problem

Input: a (finite) set $P = \{(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots (\alpha_k, \beta_k)\}$ of ordered pairs (called dominoes) of strings over Σ.

Question: Is there a match, i.e. a sequence $i_1, \ldots, i_m \in [k]$ such that $\alpha_{i_1} \cdots \alpha_{i_m} = \beta_{i_1} \cdots \beta_{i_m}$?

We allow using the same domino as many times as needed in a match.

$$\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]\left[\frac{\mathtt{b}}{\mathtt{ca}}\right]\left[\frac{\mathtt{ca}}{\mathtt{a}}\right]\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]\left[\frac{\mathtt{abc}}{\mathtt{c}}\right]$$



Post Correspondence Problem

Key idea: many-one reduction (mapping-reduction).

- Many-one reduce from $A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM accepting } w\}$ to PCP.
- As an intermediary problem we introduce a decision problem called the Modified PCP (MPCP), in which an instance of PCP is a Yes-instance iff there is a match which begins with the first domino (α_1, β_1).
- Combine two (many-one) reductions: from A_{TM} to MPCP, and one from MPCP to PCP.

Revisiting many-one reduction

Many-one reducibility: definition

Let $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$ be two languages.

We say that A is mapping-reducible (or many-one reducible) to B, written as $A \leq_m B$, if there is a computable function $f : \Sigma^* \to \Sigma^*$ such that for every input $w \in \Sigma^*$,

$$w \in A \text{ if and only if } f(w) \in B.$$

Revisiting many-one reduction

Many-one reducibility: rephrase

Let $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$ be two languages. A (many-one / mapping) reduction from A to B is an algorithm which

INPUT given an instance x to A as input

OUTPUT computes an equivalent instance y to B. That is, x is a YES-instance to A if and only if y is a YES-instance to B (i.e. $x \in A$ if and only if $y \in B$).

Revisiting many-one reduction

Many-one reducibility: rephrase

Let $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$ be two languages. A (many-one / mapping) reduction from A to B is an algorithm which

INPUT given an instance x to A as input

OUTPUT computes an equivalent instance y to B. That is, x is a YES-instance to A if and only if y is a YES-instance to B (i.e. $x \in A$ if and only if $y \in B$).

(By Church-Turing thesis,) A is mapping-reducible to B if and only if there is a reduction from A to B.

- When $A = A_{TM}$ and $B =$MPCP, we designed an algorithm which gets $x = \langle M, w \rangle$ as input and output an instance P to MPCP..

Post Correspondence Problem

Set-up

1. We assume that the TM M of instance $\langle M, w \rangle$ satisfies:
   - it is deterministic, with left/right move only.
   - M never attempts to move the header to the left when it is in the left-most cell of the tape.
   - if $w = \epsilon$, the string w is encoded as B, where B is a symbol in the alphabet.
2. Reduction from MPCP to PCP is simple:

$$\left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \left[\frac{t_3}{b_3}\right], \ldots, \left[\frac{t_k}{b_k}\right] \right\}$$

$$\left\{ \left[\frac{\star t_1}{\star b_1 \star}\right], \left[\frac{\star t_1}{b_1 \star}\right], \left[\frac{\star t_2}{b_2 \star}\right], \left[\frac{\star t_3}{b_3 \star}\right], \ldots, \left[\frac{\star t_k}{b_k \star}\right], \left[\frac{\ast \Diamond}{\Diamond}\right] \right\}$$

Chapter 5.2, Sipser 2012.

Post Correspondence Problem

Key idea for many-one reduction from A_{TM} to MPCP:

> *From an instance $\langle M, w \rangle$ to A_{TM}, create an instance (i.e. the set of dominoes) to MPCP so that there is a match if and only if <u>there is an accepting computation history of M on w</u>.*

Post Correspondence Problem

Key idea for many-one reduction from A_{TM} to MPCP:

From an instance $\langle M, w \rangle$ to A_{TM}, create an instance (i.e. the set of dominoes) to MPCP so that there is a match if and only if there is an accepting computation history of M on w.

This is an important, recurring idea: an accepting computation history is the witness / certificate that (M, w) is a Yes*-instance. And the witness of a* Yes*-instance to the target problem of a reduction emulates it.*

Post Correspondence Problem

Implementing the idea: how does a witness of MPCP, i.e. a desired match, look like?

- In a match, we essentially want to realize the string which is an accepting computation history of the form

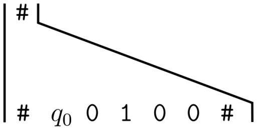
$$\#C_1\#C_2\#\cdots\#C_\ell\#$$

- In a match, the dominoes are grouped into blocks (contiguous dominoes); stage 1 depicts the initial configuration of M on w, stage 2 depicts the transitions till accept configuration, stage 3 depicts the completion of the match.

Post Correspondence Problem

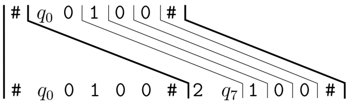
Implementing the idea: Dominoes in a desired match are grouped into stages, depending on which stage of the match it shall be used.

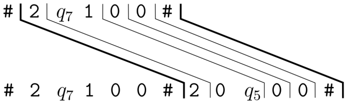
1. Stage 1: expresses a starting configuration. For this, we use the domino expressing the initial configuration.



Chapter 5.2, Sipser 2012.

2. Stage 2: expresses a transition from the config C_i to C_{i+1}.



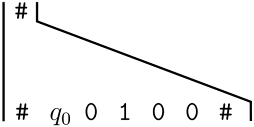


3. Stage 3: once the bottom string reaches an accept state, the dominoes let the upper string to catch up with the bottom string. (Details later.)

Post Correspondence Problem

Implementing details using "gadgets" (specifically shaped dominoes): given the instance $\langle M, w \rangle$ to A_{TM}, we progressively construct the instance P to MPCP by adding the following dominoes.

1. Gadget for Stage 1: we (the algorithm / TM) adds the domino of the form $(\#, \#q_0\ w\#)$



Chapter 5.2, Sipser 2012.

Post Correspondence Problem

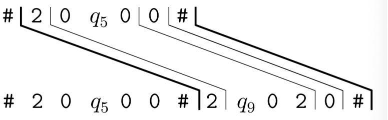
1. Gadgets for Stage 2: add dominoes for expressing the transitions as well as the tape content.
    1. Right move: For each $a, b \in \Gamma$ and each $q, r \in Q$ where $q \neq q_{reject}$, add *the domino* (qa, br) *if* $\delta(q, a) = (r, b, R)$
    2. Left move: For each $a, b, c \in \Gamma$ and each $q, r \in Q$ where $q \neq q_{reject}$, add *the domino* (cqa, rcb) *if* $\delta(q, a) = (r, b, L)$
    3. Symbol domino: for each $a \in \Gamma$, add the domino (a, a)
    4. Expressing the end of the tape content / the unused cell on the right: add the dominoes $(\#, \#)$ and $(\#, B\#)$.

Example: $\delta(q_5, 0) = (q_9, 2, L)$

$$\left[\frac{0q_50}{q_902}\right], \left[\frac{1q_50}{q_912}\right], \left[\frac{2q_50}{q_922}\right], \text{ and } \left[\frac{\sqcup q_50}{q_9\sqcup 2}\right]$$

Chapter 5.2, Sipser 2012.

$\ldots$



Post Correspondence Problem

1. Gadgets for Stage 3: add dominoes so that the upper string catches up with the bottom string once (the bottom) reaches the accept state.
    1. "Eat-up" the leftover tape content: for each $a \in \Gamma$, add the domino

$$\left[\frac{a\, q_{\text{accept}}}{q_{\text{accept}}}\right] \text{ and } \left[\frac{q_{\text{accept}}\ a}{q_{\text{accept}}}\right]$$

    2. Finish the match: add the domino

$$\left[\frac{q_{\text{accept}}\#\#}{\#}\right]$$

Post Correspondence Problem

Finishing the reduction: showing that there is a (many-one) reduction from A_{TM} to PCP consists of two parts.

1. Design a reduction. That is, we show an algorithm which maps an arbitrary instance $\langle M, w \rangle$ to A_{TM} to a suitable instance P to MPCP.
2. Establish the equivalence. we need to show that $\langle M, w \rangle \in A_{TM}$ if and only if $P \in MPCP$. That is, $\langle M, w \rangle$ is a YES-instance to A_{TM} if and only if the constructed instance P is a YES-instance to MPCP.

Post Correspondence Problem

Finishing the reduction: showing that there is a (many-one) reduction from A_{TM} to PCP consists of two parts.

1. Design a reduction. That is, we show an algorithm which maps an arbitrary instance $\langle M, w \rangle$ to A_{TM} to a suitable instance P to MPCP.
2. Establish the equivalence. we need to show that $\langle M, w \rangle \in A_{TM}$ if and only if $P \in MPCP$. That is, $\langle M, w \rangle$ is a YES-instance to A_{TM} if and only if the constructed instance P is a YES-instance to MPCP.

3. So far, we constructed a reduction (= designed an algorithm outputting an presumably equivalent instance of MPCP from an instance of A_{TM}).
4. It remains to show that the created instance of MPCP is indeed equivalent to the given input to A_{TM}.

Post Correspondence Problem

Finishing the reduction: Establishing the equivalence consists of two directions.

1. From A_{TM} to MPCP. One shows that if $\langle M, w\rangle \in A_{TM}$ (equivalently, it is a YES-instance / M accepts w), then the constructed instance P is a YES-instance to MPCP, i.e. there is a match in P beginning with the first domino.
2. From MPCP to A_{TM} One shows that if the constructed instance P allows a match starting with the first domino, then $\langle M, w\rangle$ is a YES-instance to A_{TM}.

Post Correspondence Problem

If $\langle M, w \rangle \in A_{TM}$, then P allows a match starting with the first domino.

Let $C_1, C_2, \cdots C_\ell$ be the computation history of M on w. We construct a solution of the instance P as follows.

1. Place the first domino (s_1, s_2); $s_1 = \#$, $s_2 = \#C_1\#$.
2. Use a sequence of 'symbol domino', 'transition domino', 'end-of-string/move-off-to-right domino' as needed to get

$$s_1 = \#C_1\# \qquad \text{and } s_2 = \#C_1\#C_2\#.$$

3. Continue the above 2 until you get

$$s_1 = \#C_1\#C_2\#\cdots\#C_{\ell-1}\# \qquad \text{and } s_2 = \#C_1\#C_2\#\cdots\#C_{\ell-1}\#C_\ell\#,$$

where C_ℓ is the accepting configuration.

Post Correspondence Problem

If $\langle M, w \rangle \in A_{TM}$, then P allows a match starting with the first domino.

We finish the partial match forming $s_1 = ...\#$ and $s_2 = ...\#C_\ell\#$ as follows.

4. Use the symbol dominoes and 'eat-up' dominoes for Stage 3 of the form

$$\left[\frac{a\, q_{\text{accept}}}{q_{\text{accept}}}\right] \text{ and } \left[\frac{q_{\text{accept}}\ a}{q_{\text{accept}}}\right]$$

   so that

$$s_1 = ...\#C_\ell\# \qquad \text{and } s_2 = ...\#C_\ell\#C_\ell^1.$$

   where C_ℓ^1 is a string having precisely one less symbol than C_ℓ around q_{accept}.

5. Repeat the above 4 to extend s_1 and s_2 into the forms

$$s_1 = ...\#C_\ell\#C_\ell^1\#\cdots\#C_\ell^{j-1}\# \qquad \text{and } s_2 = ...\#C_\ell\#C_\ell^1\#\cdots\#C_\ell^j\#$$

   until $C_\ell^j = q_{accept}$.

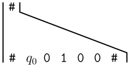
6. Use the 'finish-the-match' domino and finish the match.

$$\left[\frac{q_{\text{accept}}\#\#}{\#}\right]$$

Post Correspondence Problem

If P allows a match starting with the first domino, then $\langle M, w\rangle \in A_{TM}$ holds.
Suppose that $s_1 = s_2$ is the string formed by a match $i_1, \ldots, i_m$.

1. The match must begin with the block of the form



Chapter 5.2, Sipser 2012.

2. One should replicate C_1 in the upper string. The only dominoes containing some state are the transition domino, 'eat-up' domino, or the finish-the-match domino. Assuming $w \neq \epsilon$. there is a prefix of the form $s_1 = \#C_1\#C_2\#\cdots\#C_{\ell-1}\#$ and $s_2 = \#C_1\#\cdots\#C_2\#\cdots\#C_{\ell-1}\#C_\ell\#$, where C_i is a legal move of M from C_{i-1}.
3. Similarly, you can argue that the string from a match should be of the form $s = \#C_1\#\cdots\#C_\ell\#C_\ell^1\#\cdots\#C_\ell^j\#\#$, where C_ℓ^i omits one symbol around q_{accept} from C_ℓ^{i-1} for all $i \leq j$, and $C_\ell^j = q_{accept}$.
4. We conclude that there is a sequence of legal moves (i.e. computation history) of M on w.

Revisit: Turing-reduction from A_{TM} to REG_{TM}

$REG_{TM} = \{M : M \text{ is TM and } L(M) \text{ is a regular language}\}$.

- D, upon an input $\langle M, w\rangle$ to A_{TM}, does the following:
  1. Compute & write an encoding $\langle M_o^{M,w}\rangle$ of TM $M_o^{M,w}$ such that

$$L(M_o^{M,w}) = \begin{cases} \{0^n1^n \mid n \geq 0\} & \text{if } M \text{ does not accept } w \\ \Sigma^* & \text{if } M \text{ accepts } w \end{cases}$$

  2. Run R on $\langle M_o^{M,w}\rangle$.
  3. D outputs

$$\begin{cases} \text{YES} & \text{if } R \text{ outputs YES} \\ \text{NO} & \text{if } R \text{ outputs NO} \end{cases}$$

- Can we design a many-one reduction instead of Turing-reduction?

Revisit: Turing-reduction from A_{TM} to E_{TM}

Emptiness problem: $E_{TM} = \{M : M \text{ is TM and } L(M) = \emptyset\}$.

- D, upon an input $\langle M, w\rangle$ to A_{TM}, does the following:
  1. Compute an encoding of a TM M_o^w such that

$$L(M_o^w) = \begin{cases} \{w\} & \text{if } M \text{ accepts } w \\ \emptyset & \text{if } M \text{ does not accept } w \end{cases}$$

  2. Run the hypothetical TM E for E_{TM} on $\langle M_o^w\rangle$.
  3. D outputs

$$\begin{cases} \text{NO} & \text{if } E(\langle M_o^w\rangle) = \text{YES} \\ \text{YES} & \text{if } E(\langle M_o^w\rangle) = \text{NO} \end{cases}$$

- Can we design a many-one reduction instead of Turing-reduction?

Turing-reduction vs mapping-reduction

Emptiness problem: $E_{TM} = \{M \text{ is TM and } L(M) = \emptyset\}$.
Non-emptiness problem: $SOME_{TM} = \{M : M \text{ is TM and } L(M) \neq \emptyset\}$.

A_{TM} is Turing-reducible to E_{TM} (last week).

A_{TM} is not many-one reducible to E_{TM}.

1. Complement of E_{TM}, i.e. $SOME_{TM}$ is Turing-recognizable (how so?).

TURING-REDUCTION VS MAPPING-REDUCTION

Emptiness problem: $E_{TM} = \{M \text{ is TM and } L(M) = \emptyset\}$.
Non-emptiness problem: $SOME_{TM} = \{M : M \text{ is TM and } L(M) \neq \emptyset\}$.

A_{TM} is Turing-reducible to E_{TM} (last week).

A_{TM} is not many-one reducible to E_{TM}.

1. Complement of E_{TM}, i.e. $SOME_{TM}$ is Turing-recognizable (how so?). Think of a TM which, given $\langle M \rangle$ as input, tests all strings (in the lexicographic order) until M accepts some.

Turing-reduction vs mapping-reduction

Emptiness problem: $E_{TM} = \{M \text{ is TM and } L(M) = \emptyset\}$.
Non-emptiness problem: $SOME_{TM} = \{M : M \text{ is TM and } L(M) \neq \emptyset\}$.

A_{TM} is Turing-reducible to E_{TM} (last week).

A_{TM} is not many-one reducible to E_{TM}.

1. Complement of E_{TM}, i.e. $SOME_{TM}$ is Turing-recognizable (how so?). Think of a TM which, given $\langle M \rangle$ as input, tests all strings (in the lexicographic order) until M accepts some.

2. We know $\neg A_{TM}$ is not Turing-recognizable (why?)

3. If $A_{TM} \leq_m E_{TM}$, then $\neg A_{TM} \leq_m SOME_{TM}$, contradiction;

Turing-reduction vs mapping-reduction

Emptiness problem: $E_{TM} = \{M \text{ is TM and } L(M) = \emptyset\}$.
Non-emptiness problem: $SOME_{TM} = \{M : M \text{ is TM and } L(M) \neq \emptyset\}$.

A_{TM} is Turing-reducible to E_{TM} (last week).

A_{TM} is not many-one reducible to E_{TM}.

1. Complement of E_{TM}, i.e. $SOME_{TM}$ is Turing-recognizable (how so?). Think of a TM which, given $\langle M \rangle$ as input, tests all strings (in the lexicographic order) until M accepts some.

2. We know $\neg A_{TM}$ is not Turing-recognizable (why?)

3. If $A_{TM} \leq_m E_{TM}$, then $\neg A_{TM} \leq_m SOME_{TM}$, contradiction;
Here we use the fact that if $A \leq_m B$ and B is Turing-recognizable, then A is Turing-recognizable.