CSE 2001: Introduction to Theory of Computation

Fall 2012

Suprakash Datta

datta@cse.yorku.ca

Office: CSEB 3043

Phone: 416-736-2100 ext 77875

Course page: http://www.cs.yorku.ca/course/2001

Halting problem - recap

- Assume that it is decidable. So there is a TM S that decides

HALT=$\{\langle M,w\rangle \mid M$ is a TM and M halts on $w\}$

- Use S as a subroutine to get a TM S to decide

$A_{TM} = \{\langle M,w\rangle \mid M$ is a TM that accepts $w\}$

- Therefore A_{TM} is decidable. CONTRADICTION!
- Details follow ….

Halting problem - 2

S = “On input <M,w>

- Run TM R on input <M,w>.
- If R rejects, REJECT.
- If R accepts, simulate M on w until it halts.
- If M has accepted, ACCEPT, else REJECT”

More undecidability

$E_{TM} = \{<M>|\ M \text{ is a TM and } L(M) = \phi\}$

We mentioned that E_{TM} is co-TM recognizable.

We will prove next that E_{TM} is undecidable.

Intuition: You cannot solve this problem UNLESS you solve the halting problem!!

But this is hard to formalize, so we use A_{TM}. Instead.

E_{TM} is undecidable

Assume R decides E_{TM}. Use R to design TM S to decide A_{TM}.

- Given a TM M and input w, define a new TM M':
  - If $x \neq w$, reject
  - If $x = w$, accept iff M accepts w

S = "On input <M,w>

- Construct M' as above.
- Run TM R on input <M'>.
- If R accepts, REJECT; If R rejects, ACCEPT."

EQ_{TM} is undecidable

If this is decidable, then we can solve E_{TM}!! (You need to check equality with TM M_1 that rejects all inputs)

Assume R decides EQ_{TM}. Use R to design TM S to decide E_{TM}.

S = “On input <M>

- Run TM R on input <M, M_1>.
- If R accepts, ACCEPT; If R rejects, REJECT.”

$REGULAR_{TM}$ is undecidable

- Theorem 5.3 in the text.

The running idea

All our proofs had a common structure

- The first undecidable proof was hard – used diagonalization/self-reference.
- For the rest, we assumed decidable and used it as a subroutine to design TM's that decide known undecidable problems.
- Can we make this technique more structured?

Mapping Reducibility

Thus far, we used reductions informally:

If <u>"knowing how to solve A" implied "knowing how to solve B"</u>, then we had a **reduction** from B to A.

Sometimes we had to negate the answer to the "$\in$A?" question, sometimes not. In general, it was unspecified which transformations were allowed around the "$\in$A?"-part of the reduction.

Let us make this formal…

Computable Functions

A function $f:\Sigma^* \rightarrow \Sigma^*$ is a TM-computable function if there is a Turing machine that on every input $w \in \Sigma^*$ halts with just $f(w)$ on the tape.

All the usual computations (addition, multiplication, sorting, minimization, etc.) are all TM-computable.

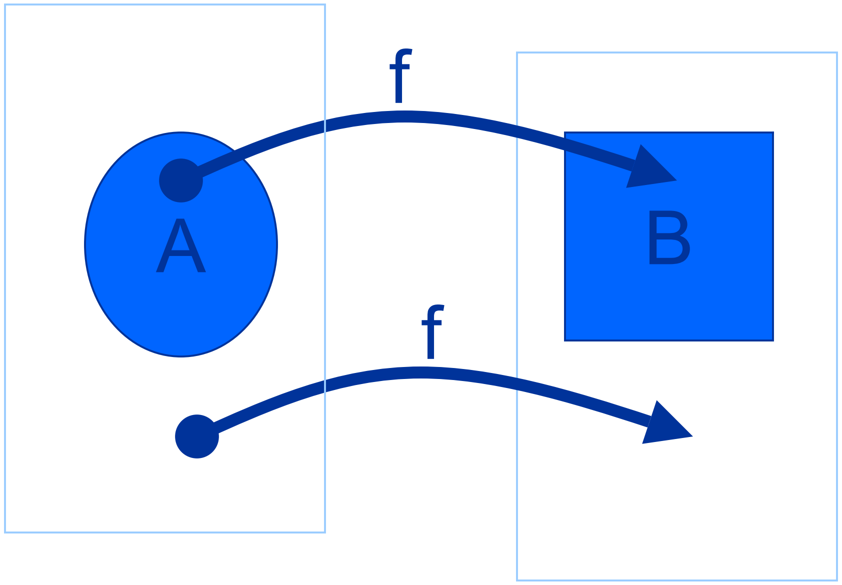
Note: alterations to TMs, like “given a TM M, we can make an M’ such that…” can also be described by computable functions that satisfy f(<M>) = <M’>.

Mapping Reducible

A language A is <u>mapping reducible</u> to a another language B if there is a TM-computable function $f:\Sigma^* \to \Sigma^*$ such that: $\mathbf{w \in A \iff f(w) \in B}$
for every $w \in \Sigma^*$.

Terminology/notation:

- $A \leq_m B$
- function f is the <u>reduction</u> of A to B
- also called: "*many-one reducible*"



$A \leq_m B$

The language B can be more difficult than A.

Intuition suggests:

Theorem 5.22: If $A \leq_m B$ and B is decidable, then A is decidable.

Corollary 5.23: If $A \leq_m B$ and A is undecidable, then B is undecidable.

Previous mappings used

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

F = “On input <M,w>

- Construct TM M’ = “on input x:
  - Run M on x
  - If M accepts, ACCEPT
  - If M rejects, enter infinite loop.”
- Output <M’,w>”

Check: f maps < M,w> to <M’, w’>.

$<M,w> \in A_{TM} \Longleftrightarrow <M',w> \in HALT_{TM}$

Previous mappings used - 2

Recall: M_1 rejects all inputs. Assume R decides EQ_{TM}. Use R to design TM S to decide E_{TM}.

S = “On input <M>

- Run TM R on input $<M, M_1>$.
- If R accepts, ACCEPT; If R rejects, REJECT.”

Check: f maps $< M>$ to $<M, M_1>$.

$<M> \in E_{TM} \Longleftrightarrow <M, M_1> \in EQ_{TM}$

Decidability obeys $\leq_m$ Ordering

Theorem 5.22: If $A \leq_m B$ and B is TM-decidable, then A is TM-decidable.

Proof: Let M be the TM that decides B and f the reducing function from A to B. Consider the TM:

On input w:

1) Compute f(w)
2) Run M on f(w) and give the same output.

By definition of f: if $w \in A$ then $f(w) \in B$.

M "accepts" f(w) if $w \in A$, and

M "rejects" f(w) if $w \notin A$.

Undecidability obeys $\leq_m$

Corollary 5.23: If $A \leq_m B$ and A is undecidable, then B is undecidable as well.
Proof: Language A undecidable and B decidable contradicts the previous theorem.

Extra: If $A \leq_m B$, then also for the complements $(\Sigma^* \backslash A) \leq_m (\Sigma^* \backslash B)$
Proof: Let f be the reducing function of A to B with $w \in A \Longleftrightarrow f(w) \in B$. This same computable function also obeys "$v \in (\Sigma^* \backslash A) \Longleftrightarrow f(v) \in (\Sigma^* \backslash B)$" for all $v \in \Sigma^*$

Recognizability and $\leq_m$

Theorem 5.28: If $A \leq_m B$ and B is TM-recognizable, then A is TM-recognizable.

Proof: Let M be the TM that recognizes B and f the reducing function from A to B. Again the TM:

On input w:

1) Compute f(w)

2) Simulate M on f(w) and give the same result.

By definition of f: $w \in A$ equivalent with $f(w) \in B$.

M "accepts" f(w) if $w \in A$, and

M "rejects" f(w)/does not halt on f(w) if $w \notin A$.

Unrecognizability and $\leq_m$

Corollary 5.29: If $A \leq_m B$ and A is not Turing-recognizable, then B is not recognizable as well.
Proof: Language A not TM-recognizable and B recognizable contradicts the previous theorem.

Extra: If $A \leq_m B$ and A is not co-TM recognizable, then B is not co-Turing-recognizable as well.
Proof: If A is not co-TM-recognizable, then the complement $(\Sigma^* \setminus A)$ is not TM recognizable.
By $A \leq_m B$ we also know that $(\Sigma^* \setminus A) \leq_m (\Sigma^* \setminus B)$.
Previous corollary: $(\Sigma^* \setminus B)$ not TM recognizable, hence B not co-Turing-recognizable .

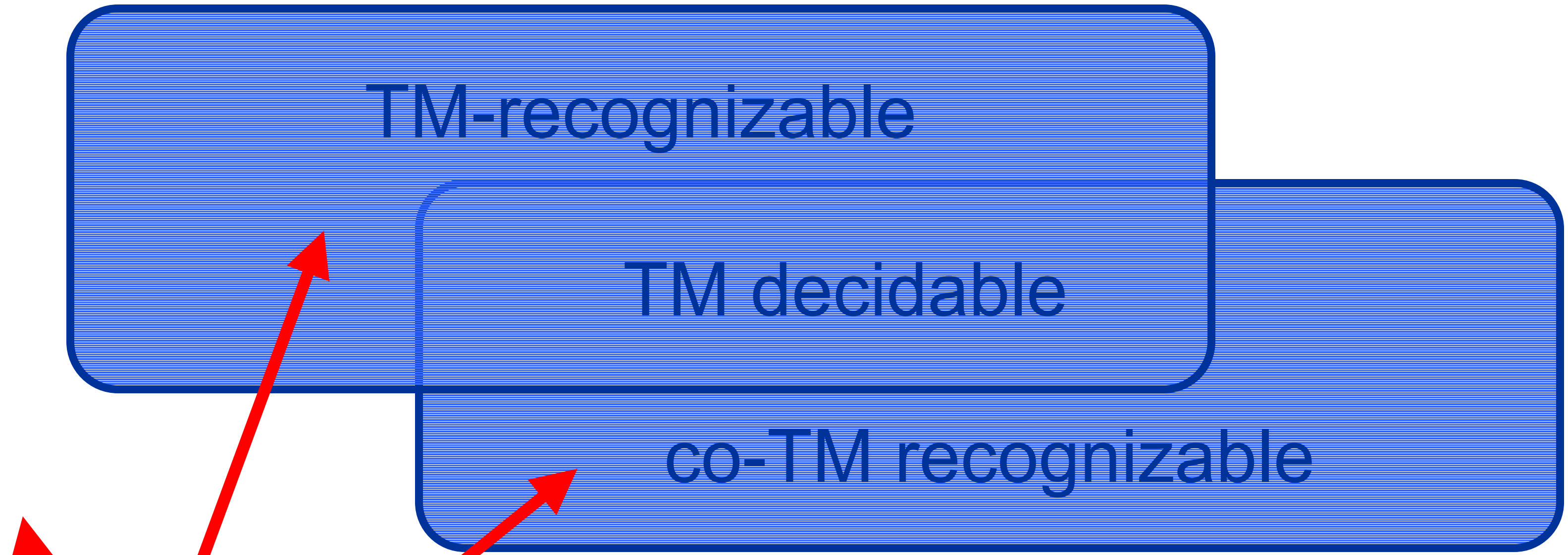
E_{TM} Revisited

Recall: The emptiness language was defined as
$E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM with } L(M)=\emptyset \}$
E_{TM} is not Turing recognizable.

Simple proof via ($\bar{A}_{TM} \leq_m E_{TM}$):
Let f on input $\langle M,w \rangle$ give $\langle M' \rangle$ as output with:
M': Ignore input
    Run M on w
    If M accepted w then "accept"
    otherwise "reject"

Now: $\langle M,w \rangle \in \bar{A}_{TM} \Longleftrightarrow f(\langle M,w \rangle) = \langle M' \rangle \in E_{TM}$

Something still unproven…



A_{TM} = { <M,w> | M is a TM that accepts w }

E_{TM} = { <M> | M is a TM with $L(M)=\emptyset$ }

EQ_{TM} = { <G,H> | G,H TMs with $L(G)=L(H)$ }

EQ_{TM} is not TM Recognizable

Proof (by showing $\bar{A}_{TM} \leq_m EQ_{TM}$):

Let f on input $<M,w>$ give $<M_1,M_2>$ as output with:

M_1: “reject” on all inputs

M_2: Ignore input

Run M on w

“accept” if M accepted w

We see that with this TM-computable f:

$<M,w> \in \bar{A}_{TM} \Longleftrightarrow f(<M,w>) = <M_1,M_2> \in EQ_{TM}$

Because $\bar{A}_{TM}$ is not recognizable, so is EQ_{TM}.

EQ_{TM} not co-TM Recognizable

Proof (by showing $A_{TM} \leq_m EQ_{TM}$):

Let f on input $\langle M,w \rangle$ give $\langle M_1,M_2 \rangle$ as output with:

M_1: "accept" on all inputs

M_2: Ignore input

Run M on w

"accept" if M accepted w

We see that with this TM-computable f:

$\langle M,w \rangle \in A_{TM} \Longleftrightarrow f(\langle M,w \rangle) = \langle M_1,M_2 \rangle \in EQ_{TM}$

Because A_{TM} is not co-recognizable, so is EQ_{TM}.

Partial $\leq_m$ Ordering