EECS 2001A : Introduction to the Theory of Computation

Suprakash Datta

Course page: `http://www.eecs.yorku.ca/course/2001`
Also on Moodle

Other Languages that are not TM-recognizable

- $E_{TM} = \{\langle G \rangle | G \text{ is a TM with } L(G) = \emptyset\}$
  - This is co-TM recognizable
    Obvious strategy: if the language is non-empty, we can find the first string that is accepted ...
  - Is it TM-recognizable (and thus decidable)?
    Answer turns out to be NO
- $EQ_{TM} = \{\langle G, H \rangle | G, H \text{ are TM's with } L(G) = L(H)\}$
  - Is this co-TM recognizable?
  - Is it TM-recognizable?
  - Turns out both answers are NO

We need more tools to reason about these languages

Easier Ways to Reason about Undecidable Problems

We will:

- Prove that the **Halting problem** is undecidable
- Do more examples of undecidable problems
- Try to get a general technique for proving undecidability

The Halting Problem

- Recall: The acceptance problem for Turing Machines:

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

  was proved undecidable “from scratch”

- What about the Halting Problem:

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

- Given the similarity to the acceptance problem, can we leverage it and get a simpler proof?
- The answer is yes....

The Halting Problem - 2

- Proof by contradiction. Suppose there is a TM H that decides $HALT$
- Main idea: Use H as a helper method to get a TM S to decide A_{TM}
- This implies that A_{TM} is decidable – Contradiction!
- Why is the acceptance problem not solvable by direct simulation? Because the simulation may never terminate!
- But H tells us if the simulation terminates, and H terminates!
- So if H says M does not terminate, M cannot accept w; if H says M terminates, then just simulate M on w

The Halting Problem - Proof Details

S on input $\langle M, w \rangle$:

- Run TM H on input $\langle M, w \rangle$
- If H rejects, REJECT
- If H accepts, simulate M on w until it halts
- If M has accepted, ACCEPT, else REJECT

Be very careful: We used the solution to an unknown problem to solve a known undecidable problem. Cannot reverse that order

The Emptiness Problem

$$E_{TM} = \{\langle M\rangle | M \text{ is a TM and } L(M) = \emptyset\}$$

- We showed 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

Note: We now have 2 provably undecidable problems and can leverage either

E_{TM} is Undecidable

- Proof by contradiction. Suppose there is a TM R that decides E_{TM}
- Main idea: Use R as a helper method to get a TM S to decide A_{TM}
- Very clever construction:
  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
- Idea: M' is empty iff M accepts w

E_{TM} is Undecidable - 2

The machine S that decides A_{TM} is as follows
On input $\langle M, w \rangle$

- Construct M' as in the last slide
- Run TM R on input $\langle M' \rangle$
- If R accepts, REJECT
  Else If R rejects, ACCEPT

EQ_{TM} is Undecidable

$$EQ_{TM} = \{\langle M_1, M_2\rangle | M_1, M_2 \text{ are TM's and } L(M_1) = L(M_2)\}$$

- Idea: 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 $\langle M\rangle$

- Construct M_1 that rejects every input
- Run TM R on input $\langle M, M_1\rangle$
- If R accepts, ACCEPT;
  Else If R rejects, REJECT

Summary of Techniques Used to Prove Unidecidability

- 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
- Q: Can we make this technique more structured?
- We still have not shown that EQ_{TM} is not TM-recognizable, and that EQ_{TM} is not co-TM-recognizable

EQ_{TM} is Not TM-Recognizable

- What can we use?
  Not much choice, except $\overline{A_{TM}}$, E_{TM}
- So we have to show if we can build a recognizer for EQ_{TM}, we can build a recognizer for E_{TM}
- This is a contradiction
- So EQ_{TM} is not TM-recognizable
- Intuition: If we have a recognizer for checking equality, we can use it to recognize equality with a TM that rejects everything

EQ_{TM} is Not TM-Recognizable - Details

- Proof by contradiction: Assume EQ_{TM} is TM-recognizable, and there is a recognizer R for it
- Given R, build a recognizer S for E_{TM} as follows
  - Construct (the description of) a machine M_e that rejects all inputs
  - Take the input machine M of E_{TM} and construct input $\langle M, M_e \rangle$ for EQ_{TM}
  - Run R on $\langle M, M_e \rangle$
    If R accepts, ACCEPT Else if R rejects, REJECT
- Note that S is not guaranteed to halt because R may not halt - that is ok since we are building a recognizer

EQ_{TM} is Not TM-Recognizable - Alternative Proof

Let us use $\overline{A_{TM}}$ instead

- Proof by contradiction: Assume EQ_{TM} is TM-recognizable, and there is a recognizer R for it
- Given R, build a recognizer S for $\overline{A_{TM}}$ as follows
  - Construct a machine M_e that rejects all inputs
  - Take the input $\langle M, w \rangle$ of $\overline{A_{TM}}$ and construct a TM M' that ignores its input, runs M on w and ACCEPTS if M accepted w
  - Construct input $\langle M', M_e \rangle$ for EQ_{TM}
  - Run R on $\langle M', M_e \rangle$
    If R accepts, ACCEPT Else if R rejects, REJECT
- Crucial fact: S accepts iff M does not accept w
- Note: again S is not guaranteed to halt because R may not halt

EQ_{TM} is Not co-TM-Recognizable

- Let us use A_{TM}
- Proof by contradiction: Assume EQ_{TM} is co-TM-recognizable, and there is a recognizer R for it (R always halts and rejects if the inputs are unequal)
- Given R, build a recognizer S for A_{TM} as follows
  - Construct a machine M_a that **accepts** all inputs
  - Take the input $\langle M, w\rangle$ of A_{TM} and construct a TM M' that ignores its input, runs M on w and ACCEPTS if M accepted w
  - Construct input $\langle M', M_a\rangle$ for EQ_{TM}
  - Run R on $\langle M', M_a\rangle$
    If R accepts, ACCEPT Else if R rejects, REJECT
- Crucial fact: S accepts iff M accepts w
- Note: again S is not guaranteed to halt because R may not halt

Enumerability and Recognizability

- Terminology: Recursive or decidable, (recursively) enumerable or recognizable
- Crucial fact: The set of all enumerable languages is countable
- How to enumerate an enumerable language?
- Straightforward idea: enumerate all strings, see if recognizer accepts it
- Problem: recognizer may not halt!
- Next idea: run for 1 step on all inputs, then 2 steps on all inputs,...
- Problem: there are infinitely many inputs!

Enumerating a Recognizable Set - Solution

Really smart idea (Page 179 of the text)!

- Simulate recognizer for 1 step on input 1
- Simulate recognizer for 2 steps on inputs 1, 2
- Simulate recognizer for 3 steps on inputs 1, 2,3
- Simulate recognizer for i steps on inputs $1, 2, \dots, i$
- Will accept inputs in increasing order of steps, not indices
- Each time an input is accepted, write it on a tape – enumeration