EECS 2001N: Introduction to the Theory of Computation

Suprakash Datta
Office: LAS 3043

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

Recap from Last Lecture

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