COMS3261: Computer Science Theory

Fall 2013

Mihalis Yannakakis

Lecture 22, 11/25/13

Halting Problem

- Input: (Code of) Turing machine M, input string w
- Question: Does M halt on input w?
- Halting problem is undecidable
- The language L_h = { <M,w> | TM M halts on input w} is r.e. but not recursive.
- To show L_h is r.e.: Simulate M on input w (like the universal machine) and accept if M halts.
- To show it is not recursive: Show that the complementary language L_h^c = { <M,w> | TM M does not halt on input w} is not r.e.
- Reduction from the complement of the universal language, L_u^c = {<M,w> | TM M does not accept w }

$L_u^c \leq L_h^c$

Given an input x=<M,w> for the first problem, compute an input f(x)=<M' ,w'> for the second problem such that

$$x \in L_u^c \Leftrightarrow f(x) \in L_h^c$$

- M' = modification of the TM M where whenever M halts at a rejecting state for some tape symbols (i.e. does not have a transition), M' instead loops forever; and whenever M is at an accepting state, M' halts.
- w'=w

For any input w, M' follows the same computation as M, and
- if M accepts then M' halts
- if M does not accept then M' runs forever

Emptiness

- L_e = { M | L(M) = ∅ } not RE
- L_{ne} = { M | L(M) ≠ ∅ } RE but not recursive
- Proof:
- L_{ne} is RE: Nondeterministic TM that accepts L_{ne}: Guess a string w, i.e. nondeterministically add one more symbol to w or terminate it. Then verify that M accepts w.

  \- Deterministic TM: Consider the enumeration of all strings. Compute in rounds 1,2 ….

  Round i: Simulate M for i steps on each of the first i strings.

  \- If M accepts some string, say w_i, and takes n steps , then TM will accept in round max(i,n)

Emptiness ctd

- L_{ne} is not recursive (L_e not RE)
- Reduce L_u to L_{ne} (L_u^c to L_e)

- Given M,w compute a TM M' such that M accepts w iff $L(M') \neq \varnothing$.
- M': takes input x, which it ignores and just simulates M on w; if M accepts w, then M' accepts x, if not, then not.

- M accepts w $\Rightarrow$ L(M')= Σ^*
- M does not accept w $\Rightarrow$ L(M')= $\varnothing$

Comparing 2 TMs

- EQ={ <M1,M2> | L(M1)=L(M2) } is not RE and neither is its complement: INEQ ={ <M1,M2> | L(M1)$\neq$L(M2) }
- Proof: Both parts by reduction from L_u^c
- Same reduction from L_u^c as for emptiness:
- Given M,w, let M' be the TM that for any input x, it ignores it, runs M on w and accepts if M accepts
- 1. EQ not RE: M does not accept w iff <M',N> $\in$EQ where N is the TM that accepts $\varnothing$.
- 2. INEQ not RE. M accepts w iff <M',N'> $\in$EQ where N' is the TM that accepts Σ^*,

  i.e. M does not accept w (<M,w>$\in$ L_u^c) iff <M',N'> $\in$INEQ

Implications for Programs

- TM's a formal stand-in for programs.
- General-purpose programming languages (C, C++, Java…) are Turing-complete (can compute anything that TMs can)
- Similar questions for programs, as for TMs, are undecidable e.g. Does a program halt on a given input? Does it halt on all inputs? Does it print "hello, world"? Does it ever execute a particular statement?, Does a variable ever become 0? …
- 2-Counter machines are equivalent to TMs, so undecidable questions hold for `simple' programs that use just one enumerated variable (for the state) and two integer variables (for the counters) with operations ++, -- and test for 0.

Properties of languages

- A property of RE languages = a set of recursively enumerable languages
- A property is trivial if it is either empty (no RE language satisfies it) or is all RE languages; otherwise nontrivial
- Examples of nontrivial properties:
- "$=\emptyset$" , "$\neq\emptyset$"
- "$=\{0,1\}^*$"
- "regular", "nonregular"
- "CFL"
- "finite", "infinite"
- "contains ε"
- "contains 001"

Rice's Theorem

- Rice's Theorem: For every nontrivial property P of RE languages, $L_P = \{ \langle M \rangle \mid L(M) \text{ satisfies } P\}$ is undecidable

- Note (important): Theorem concerns a property of the language of the TM, not of the TM itself.

  e.g. TM M has 100 states, is a property of the TM (not the language) and is actually decidable

- The theorem tells us that L_P nonrecursive. This implies that either L_P is not recursively enumerable, or its complement is not r.e. or neither (because if both L_P and its complement were r.e. then L_P would be recursive). However, the theorem as stated does not say which of these is the case.

Proof of Rice's Theorem

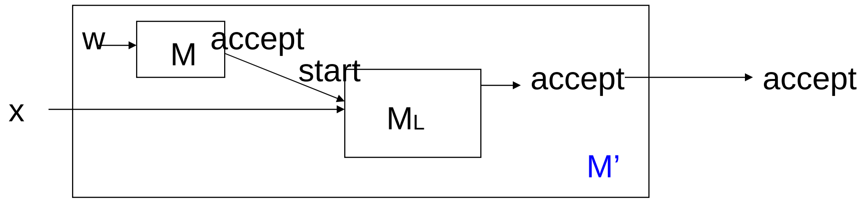
Reduction from Lu

- Assume $\varnothing$ does not satisfy P, otherwise consider the complementary property P^c and $L_P{}^c$
- Reduction from L_u to L_P
- Let L be a (non $\varnothing$) RE language that satisfies P and M_L a TM that accepts L.

- Given M,w input for L_u, construct TM M' (=$M'_{(M,w)}$) such that M accepts w iff L(M') satisfies P.
- In particular, $(M,w) \in L_u \Rightarrow L(M')=L$, satisfies P
- $(M,w) \notin L_u \Rightarrow L(M')= \varnothing$, does not satisfy P

- $L_u{}^c$ is not RE $\Rightarrow$ Proof shows $L_P{}^c$ is not RE if $\varnothing$ does not satisfy P, and L_P is not RE if $\varnothing$ satisfies P

Proof of Rice's Theorem



- M' : Run M on w.
- If M accepts w, then { run M_L on input x;
  if M_L accepts x, then accept, else reject }
- M' rejects if M rejects on w (may run forever) or M_L rejects x
- $(M,w) \in L_u \Rightarrow L(M')=L \Rightarrow L(M')$ satisfies P
- $(M,w) \notin L_u \Rightarrow L(M')=\varnothing \Rightarrow L(M')$ does not satisfy P