CSE 396 Lecture Thu 4/14 Spr 2016

<u>Theorem</u>: For every program P that can be written in any high-level language that has ever been devised, we can find a Turing machine M that is <u>equivalent</u> to P in all of several senses:

1. <u>If</u> we consider the language L(P) of your P to be $\{x \in ASCII^* : P(x) \text{ exits with error status } 0\}$ then $L(M) = L(P)$.

   Or "1.5" – if we consider M to <u>compute</u> a characteristic <u>function</u>: $L(M) = \{x : M(x) \text{ halts and outputs } 1\}$.

2. If P computes a numerical function, then M can be a TM <u>transducer</u> that computes the same fn.
3. M emulates every "step" of P by step(s) of its own.

<u>Consequence for us</u>: • TMs are a generally relevant hard-and-fast model for assessing real programs.

• It is henceforth OK to describe TMs in pseudocode.

<u>Theorem</u>: For every NTM N we can build a DTM M s.t. $L(M) = L(N)$.

Proof ↑ Text does more directly.

• Write in Java a program P that tries all possibilities for N on a given x and accepts iff when an accepting comp^n of N on x is found. ∴ $L(P) = L(N)$ • Convert P to DTM M. ☒

(2)

☆ The classes $\underline{RE} = \{L : L = L(M)$ for some DTM $M\}$

and $\underline{REC} = \{L : L = L(M)$ for some $\underline{\text{total}}$ DTM $M\}$

are the same with NTMs ~~or~~ Java or any other known program concept.

Hence the notions of recognizable and decidable apply generally to (ie. languages) $\underline{\text{decision problems}}$ (and also functions)

Some examples, in "INSTANCE - QUESTION" format:

← The text avoids this much-used format in order that names like "A_{DFA}" stand only for languages. This avoids temporary stacked subscripts in languages like $L_{A_{TM}}$

· $\underline{\text{ACCEPTANCE PROBLEM FOR DFAs } (A_{DFA})}$

INST: A DFA M, an argument x to M.
QUES: Does M accept x?

Easily Decidable — Just run $M(x)$.

Angle brackets ≡ some sensible encoding over ASCII

The language $L = L_{A_{DFA}}$ of this problem is $\{\underbrace{\langle M, x\rangle} : M \text{ accepts } x\}$

Note: $M = (Q, \Sigma, \delta, s, F)$ say $\Sigma = \{a, b\}$ a single string w $\underline{\text{over ASCII}}$.

$L(M)$ itself is over Σ.

But $L_{A_{DFA}}$ is over ASCII since it handles M's for "any" Σ.

(Ultimately, theoretically, we blur all distinctions and consider all languages to be over the alphabet $\Sigma = \{0, 1\}$.)

2. (Non-)EMPTINESS PROBLEM FOR DFAs. (NE_{DFA})

$\underline{\text{INST}}$: Just a DFA $M = (Q, \Sigma, \delta, s, F)$

QUES: Is there any $x \in \Sigma^*$ s.t. M accepts x. I.e. Is $L(M) \neq \emptyset$?

2: EMPTINESS FOR DFAS (E_{DFA})
INST: A DFA M
QUES: Is $L(M) = \emptyset$?

These problems are "equally decidable" since a yes answer for one $\equiv$ a no answer to the other. ③.

Thm: Both of these problems are decidable by Breadth First Search.

NE_{DFA}: Given M, $L(M) \neq \emptyset \iff$ there exists a path from s to some final state f.

We can code BFS into a program P. P is total since BFS always terminates. $\therefore$ Can convert P to a total DTM M_T.

[Whatever chars appear on the path form an accepted string x.]

$LNE_{DFA} = L(M_T)$ and M is total, so NE_{DFA} is also called decidable.

$M_T = (Q, \Sigma, \Gamma, \delta, B, s, \{q_{acc}, q_{rej}\})$

$M'_T = (Q, \Sigma, \Gamma, \delta, B, s, \{q_{rej}, q_{acc}\})$ is likewise total, and $L(M'_T) = LE_{DFA}$.

Because M_T is total,

Theorem: The class REC is closed under complements.

$\equiv$

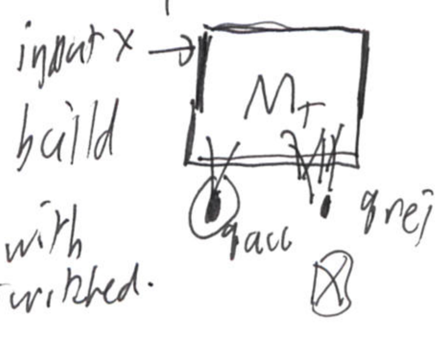
For every language L that is decidable, $\tilde{L}$ is also decidable.

$\equiv$

For every total TM M_T we can build a total TM M'_T s.t. $L(M'_T) = \widetilde{L(M_T)}$.

$\equiv$

For every total TM M_T we can build M'_T such that for all $x \in \Sigma^*$, M'_T accepts $x \iff M_T$ does not accept x.



Proof: Given $M_T = (Q, \Sigma, \Gamma, \delta, B, s, q_{acc}, q_{rej})$ build M'_T as above.

M'_T is the same box with q_{acc} and q_{rej} switched.

4. ~~ACC~~ ACCEPTANCE FOR TMs (A_{TM})

INST: A DTM M, an input $x \in \Sigma^*$ to M.
QUES: Does M accept x?

⚠ Unlike as we did for DFAs, we can't decide this so easily by "Just run $M(x)$" because $M(x)$ might never halt.

Q: Is there a clever way like with BFS for DFAs?

<u>An "Easier" Problem</u> Self Acceptance for TMs (K_{TM}) standard, not in text.

INST: A DTM M.
QUES: Does M accept $\langle M \rangle$? ← some fixed encoding of TMs M as string(s).

The complement of the problem is <u>D_{TM}</u> [special case of A_{TM} where $x = \langle M \rangle$.]

INST: A DTM M
QUES: Does M <u>not</u> accept $\langle M \rangle$?

How text <u>would</u> name it, as a language without writing $L_{D_{TM}}$ ↓ Not to confuse with a DTM.

$L_{D_{TM}}$, i.e. just the language $\boxed{D_{TM}, = \{\langle M \rangle : M \text{ does not accept } \langle M \rangle\}}$.

This and K_{TM} are "equally decidable", but neither <u>is</u> decidable.

In fact, $L_{D_{TM}}$ is not even recognizable. <u>Proof</u>:

Suppose $L_{D_{TM}}$ were {c.e. / recognizable / r.e.}. Then there would be a deterministic TM Q such that $L(Q) = L_{D_{TM}}$. I.e. s.t. $\forall \langle M \rangle$: Q accepts $\langle M \rangle \Leftrightarrow M \in L_{D_{TM}}$

In particular, <u>Q accepts $\langle Q \rangle$</u> $\Leftrightarrow \langle Q \rangle \in L_{D_{TM}} \Leftrightarrow$ <u>Q does not accept $\langle Q \rangle$</u>

In logic, a statement S may never $\Leftrightarrow$ its negation $\neg S$. ∴ Q cannot exist ∴ $L_{D_{TM}}$ is not ∈ RE.