CSE 396 Lecture Thu Apr 19 "Spr" 2018

Top Hat 8053

Sipser's Problem Naming Scheme:

A_{DFA}

A — Kind of Question

DFA — category of machine or other <u>formal</u> <u>object</u> being asked about.

A: "Does <u>M</u> accept <u>x</u>?" = accept. Two givens

E: "Is $L(\underline{M}) = \emptyset$" E for empty? Has just one given: an encoding of a machine M

<u>NE</u>: "Is $L(\underline{M}) \neq \emptyset$?" NE for <u>non-emptiness</u>: $\underbrace{(\exists x)}_{x \text{ is quantified: } \underline{\text{not}} \text{ a given}} x \in L(M)$?

ALL: Is $L(M) = \Sigma^*$?"

<u>Empty ∩</u>: Given <u>M_1</u> and <u>M_2</u>, is $L(M_1) \cap L(M_2) = \emptyset$?

<u>EQ</u>ual: Given M_1 and M_2, is $L(M_1) = L(M_2)$?

The above problems talk only about accept/reject and can be worded the same to be about regexps <u>r</u> or grammars <u>G</u>. More specific to machines, we can ask:

HALT (HP): Given <u>M</u> and <u>x</u>, does <u>$M(x)\downarrow$</u>?

<u>TOT</u>: Given just <u>M</u>, does M halt for all inputs? i.e., does $(\forall x\, M(x)$ — i.e. $(\forall x) M(x)\downarrow$?

How problems are specified:

<u>A_{DFA}</u> — <u>Name</u> of problem, often synonymous with the language L of the problem

INPUT: A DFA <u>M</u> and an <u>x</u> $\in \Sigma^*$

QUERY: Does M accept x?

<u>Language</u> is

$L = \{ z = \overline{\langle M, x \rangle}$: the answer is <u>yes</u>, i.e. M accepts x $\}$

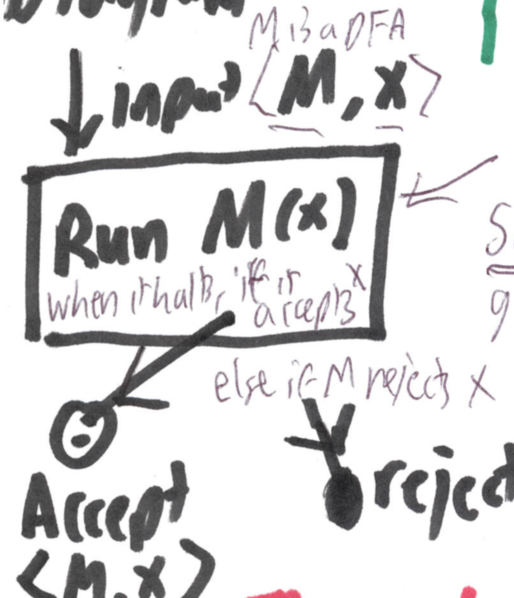
Algorithm to decide A_{DFA} problem/lang.

Pseudocode for a TM or code

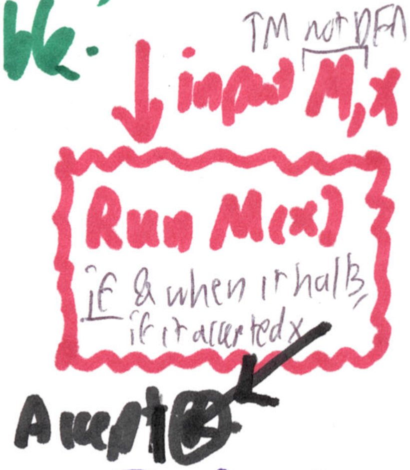
0. Given DFA M and $x \in \Sigma^*$ $(M = (Q, \Sigma, \delta, s, F))$
1. Run M on x. M must halt within $|x|+1$ of its own steps.
2. If M accepts x, you accept $\langle M, x \rangle$. If not, you reject $\langle M, x \rangle$.

Since your algorithm is ~~are~~ total, the A_{DFA} language / problem is decidable.

Flowchart Diagram



old Flowchart Convention: Solid box means routine is guaranteed to halt. Fuzzy box ≡ might not exit.



$A_{TM} = \{\langle M, x \rangle : M \text{ is a Turing Machine and } M \text{ accepts } x\}$

The code in both cases can be the "Turing Kit". Turing Kit is not a decider — not total — but does show that the A_{TM} language is computably enume, i.e. c.e.

We will see next week that A_{TM} is not in DEC.

But A_{DFA} is decidable, because when M is a DFA, both M and Turing Kit on M are guaranteed to halt.

A_{DPDA} : INPUT: A DPDA $M = (Q, \Sigma, \ldots)$ and an $x \in \Sigma^*$ ↓ M, x

QUES: Does M accept x? (∗)

First edit M to M′ which makes all non-halting $[q, {c_1 \atop c_2}/\ldots]$ cases go straight to q_{rej}. Then Run $M'(x)$ if it accepts, else

Accept

reject

Hardware (∗): Yes, this edit can always be determined, so algorithm is total.

A_{NFA} :

INST: An NFA N, an $x \in \Sigma^*$

QUES: Does N accept x?

Can convert N to an equiv't DFA M, then run M(x).

but this can incur exponential blowup in time.

Can solve in poly(|x|) time by simulating N(x) directly keeping track of which states are "currently lit".

This is how UNIX grep and scripting langs solve A_{REGEX}

NE_{NFA} : INSTANCE: An NFA N (no "x" this time)

QUESTION: Is $L(N) \neq \emptyset$?

$N = (Q, \Sigma, \delta, s, F)$

$L(N) \neq \emptyset \iff$ N has a path from s to some state in F.

The $x \in L(N)$ is whatever chars are processed on that path.

Algorithm:
- Do Breadth-First Search starting from s. Must halt within m iterations, $m = |Q|$.
- Accept ⟨N⟩ iff some state in F is found.

This is a decider for NE_{DFA} too.

How about ALL_{DFA}: $\underline{I}$: A DFA $M=(Q,\Sigma,\delta,s,F)$

Q: Is $L(M)=\Sigma^*$?

Decider:

1. Given M, first convert M into M' s.t. $L(M') = \sim L(M)$ i.e. $\Sigma^* \setminus L(M)$.

$M' = (Q,\Sigma,\delta,s,\underline{Q\setminus F})$ runs in poly time.

Hence $L(M)=\Sigma^* \Leftrightarrow L(M')=\emptyset \Leftrightarrow \langle M'\rangle \notin NE_{DFA}$ ie. $\in E_{DFA}$

2. Feed $\langle M'\rangle$ to your algorithm for NE_{DFA}. if it accepts, you reject; if it rejects you accept.

So this is a correct decider. [Idea for Ch 5: we reduced ALL_{DFA} to the NE_{DFA} problem

How about ALL_{NFA}?

Possible decider in flowchart form.

↓ input N (an NFA)

$\boxed{\text{Convert } N \text{ to equiv DFA } M}$

↓

$\boxed{\text{run above alg for } ALL_{DFA}}$

Decider, but not in poly(n) time!

In fact ALL_{NFA} and ALL_{REGEXP} are both NP-hard.

How about ALL_{DPDA}?

Can use same complementing trick after converting M to DPDA M that always halts. Then run NE_{CFG} algm → next lecture.

How about ALL_{NPDA} and ALL_{CFG}?

The trick can't be made to work.

In fact, THERE IS NO DECIDER!