CSE 396 | Lecture Tue May 7 | Spring 2019 (1)

Top Hat 6013

What kind of machine (or machine duo) can **verify** computations by a (possibly nondeterministic, 1-tape) Turing Machine $M = (Q, \Sigma, \Gamma, \delta, \sqcup, s, q_{acc}, q_{rej})$ on an input x?

finite

(possible) Computation (path) $\vec{c} = [I_0(x)][I_1][I_2] \cdots [I_j][I_{j+1}] \cdots [I_t]$

I_t: wants to be a halting ID

Prototypical of checking a proof

Checking $I_j \vdash_M I_{j+1}$ is mostly like checking equality of two strings (over $Q \cup \Gamma \cup \{[,]\}$ Σ!

$I_0(x) = sx$ general $I_j = uqcv$ acc ID q_{acc} | by G.R.

Except we need to check I_{j+1} follows from a legal instruction in δ, but this is a very "local edit".

If we write odd IDs in reverse, then $I_j \vdash I_{j+1}$ is mostly like palindromes.

marked by][brackets.

$\vec{c}' = [I_0(x)][I_1^R][I_2][I_3^R][I_4][I_5^R] \cdots [I_t^{(R)}]$

A duo of two DPDAs can check this. (D_1 and D_2)

D_1: push, pop. Check transition from δ on the fly in finite control.

D_2: Second DPDA

D_1, D_2, D_1, D_2

whichever: Both check whether last ID is halting.

∴ The language $VC(M)$ of valid accepting computations by a TM M is the intersection of two DCFLs. (halting)

The complement $\sim VC_M$ **is** a CFL, roughly because ① we only need to check failure of $I_j \vdash_M I_{j+1}$ in one place (Guess a j)

and checking $I_j \not\vdash_M I_{j+1}^R$ (or with I_j^R) is like the complement of PALindromes which is a CFL, with grammar G!

★ $M \in E_{TM} \iff \underbrace{VC_M = \emptyset}_{\text{M has no valid accepting computations, not on any input } x.} \iff \widetilde{VC}_M = \Sigma^* \iff L(D_1) \cap L(D_2) = \emptyset$

$\iff L(G') = \Sigma^*$

This is the correctness condition for reductions from E_{TM} to these two problems.

① $DCFL_{E\cap}$: INST : Two DPDAs D_1, D_2
DPDA empty intersection QUES : Is $L(D_1) \cap L(D_2) \neq \emptyset$

② ALL_{CFG} $\equiv_m$ ALLPDA : INST : A CFG G'
QUES : Is $L(G') = \Sigma^*$?
(where we could re-code Σ' over $\Sigma = \{0,1\}$, say).

The last fact is that D_1, D_2, and G' can be computed given only the code $\langle M \rangle$ of M.

Thus $E_{TM} \leq_m DCFL_{E\cap}$ and $E_{TM} \leq_m ALL_{CFG}$, so these languages are undecidable, indeed not c.e.

Emptiness and $ALL_{(\ldots)}$ are likewise undecidable for any class of automata or grammars or expressions capable of recognizing VC_M. Two examples:

- Linear Bounded Automata (LBAs) $\equiv$ TMs that can change only the n cells initially occupied by the input x.
- Two-Head DFAs which have 2 tapes and begin with x on both.

These are LBAs and unlike LBAs they run in $O(N)$ time, hence in polynomial time. N here $= |\vec{c}|$

2HDFAs capture the idea of the Post Correspondence Problem in the skipped §5.

Defn: A DTM or NTM M runs in polynomial time if for all $x \in \Sigma^*$,

$x \in L(M) \Rightarrow M(x)$ has an acc computation with $t \leq p(n)$, where p is some polynomial function and $n = |x|$.

$x \notin L(M) \Rightarrow M(x)$ has no acc comps at all but halts within (NTM: every computation path halts in $p(n)$ steps.)

Fact: Every alg^m that runs in $O(n^c)$ time in the CSE331 modeling runs in $O(n^{c'})$ time on a TM, where $c' \leq 4c$. ($8c$ for 1-tape final TM)

Hence these classes have the same defn for TMs as for RAMs and HLLs.

$P = \{L(M) : M$ runs in poly time, M is a DTM$\}$

$NP = \{L(N) : N$ is an NTM that runs in poly time$\}$

Thm (Ch3): For every NTM N we can build a DTM M s.t. $L(M) = L(N)$.

Proof: $N \hookrightarrow$ Turing Kit by simulation maintaining a data structure of all computation branches, looping over 1-step updates of each one (if-and) - until you find that some branch accepts $\hookrightarrow$ Java $\hookrightarrow$ DTM M from the Univ. RAM simulator.

Problem: M will still take exponential(n) time from this.

Central Question: Can we do it faster? $\equiv P = NP$?

Theorem: A language L belongs to NP if and only if there is a polynomial $p(n)$ and there is a polynomial time decidable language R in P s.t. for all $x \in \Sigma^*$, $x \in L \iff (\exists y) : |y| \leq p(|x|) \wedge R(x, y)$.

Proof: Take N s.t. N accepts L and runs in poly time $q(n)$.

Then $x \in L \iff \exists \vec{C} : \underbrace{\vec{C} \text{ has an acc comp of } N \text{ on input } x}$, verify this with a 2HDFA, which runs in $O(|\vec{C}|)$ time, and $|\vec{C}| \leq \underbrace{q(n)^2}_{= \text{ the } p(n)}$

Text: (the 2HDFA) is a poly-time verifier

Since: $\vec{C} \leq q(n)$ steps times max size of any ID I_i in $\vec{C}$.

Added: Conversely, given a poly-time decider M_R for $R(x,y)$, we can build an NTM N that on input x guesses y and then verifies $R(x,y)$. ☒ Since $|y| \leq p(|x|)$ and poly(poly(n)) = poly(n), N runs in poly time, so $L \in NP$.

The Ch4 deciders for E_{DFA}, E_{NFA}, ALL_{DFA}, (EQ_{DFA}), E_{CFG} all run in poly time.

A_{CFG} is in $\mathbf{P}$ for reasons not in Ch4. So every CFL is in P.