CSE 396 Review Session 5/15/17 ①

Source Problem: A_{TM}

INST: A TM M and an input w to M $\langle M, w\rangle$

QUES: Does M accept x?

Target Problem: EXC THROW

INST: A pgm P and an input x to P. Type $\langle P, x\rangle$

QUES: Does $P(x)$ throw an ARR excep^n?

Answer in Ch 4, PS 1
mindset: Suppose EXC THROW were decidable by a total program S (or Turing Machine) (S is a decider)

By defn of decider S can correctly decide all cases, even "silly" ones of the form

$P_{M,w}$ = input x (ignored) (could be $x = \langle M, w\rangle$, not required)
{ Simulate $M(w)$ using a Java TM simulator [Turing Kit, JFLAP, or etc.]
if & when it accepts
THROW ARRAYIND...EXC }

Thus we would get a decider R for the A_{TM} problem by:

1. Given $\langle M, w\rangle$, build $P_{M,w}$
2. Run S on $\langle P_{M,w}, \lambda\rangle$ — nothing wrong with putting $\langle M, w\rangle$ again here instead
3. Accept $\langle M, W\rangle$ iff S accepts $\langle P_{M,w}, \lambda\rangle$

Why is R correct?

$\langle M, W\rangle \in A_{TM} \equiv M$ accepts $w \Rightarrow$ on any x, certainly $x = \lambda$, $P_{M,W}(x)$ sees the accept and throws the exception $\Rightarrow S$ accepts $\langle P_{M,W}, \lambda\rangle$ since S is a decider for all cases of the target problem, even silly ones $\Rightarrow R$ accepts $\langle M, W\rangle$ by how R is coded.

$\langle M, w\rangle \notin A_{TM} \Rightarrow M$ does not accept $w \Rightarrow$ on any x (certainly $x = \lambda$), $P(x)$ does not throw the exception there, nor anywhere because simulator never throws it $\Rightarrow S$ does not accept $P_{M,W} \Rightarrow R$ does not accept $\langle M, W\rangle$.

$\therefore R$ accepts $\langle M, W\rangle \iff \langle M, W\rangle \in A_{TM}$ so $L(R) = A_{TM}$ and R halts for all inputs, so A_{TM} is decidable. But this is a contradiction.

Define $f(\langle M, W\rangle) = \langle P_{M,W}, \lambda\rangle$. This computable fn gives $A_{TM} \leq_m$ Targ Problem. $\langle M, W\rangle \in A_{TM} \iff f(\langle M, W\rangle) \in L_{Targ\ Problem}$. ☒

2) $\langle M, w\rangle \in A_{TM} \Rightarrow$ on any x, $P_{M,w}(x)$ executes ~~throws~~ System.exit(0) 2

$\boxed{\text{yes accept} \Downarrow \text{yes } L(P_{M,w}) \text{ is nonempty.}}$ $\Rightarrow L(P_{M,w}) = \Sigma^* \Rightarrow P_{M,w}$ accepts its own (Java) code

$\langle M, w\rangle \notin A_{TM} \Rightarrow$ on any x, $P_{M,w}(x)$ does not reach the accept/throw

$\Rightarrow L(P_{M,w}) = \emptyset \Rightarrow P_{M,w}$ does not accept its own code.

$\therefore A_{TM} \leq_m K_{Java}$

Also $A_{TM} \leq_m ALL_{Java}$ and $A_{TM} \leq_m NE_{Java}$.

$\therefore K_{Java} \leq_m A_{TM}$ by $f(P) = \langle P, P\rangle$ instance of A_{Java}

$= \langle M, M\rangle$ where M is a TM that simulates P and decompiles $\langle M\rangle$ back into $\langle P\rangle$.

$A_{TM} \leq_m K_{TM}$ similarly
$K_{TM} \leq_m A_{TM}$ via $f(M) = \langle M, M\rangle$.

3) Source E_{TM}: Inst "M"
Ques: Is $L(M) = \emptyset$?

Target: $REGULAR_{CFG}$
Inst: A CFG G
Ques: Is $L(G)$ ~~nonempty~~ regular?

To reduce: define $f(M)$ = the G s.t. $L(G) = \sim ACH_M$.

Fact: $\Longleftrightarrow ACH_M = \emptyset$.

$\langle M\rangle \in E_{TM} \Rightarrow L(G) = \Sigma^* \Rightarrow L(G)$ is regular $\Rightarrow f(M) \in REG_{CFG}$.

$\langle M\rangle \notin E_{TM} \Rightarrow ACH_M \neq \emptyset \xRightarrow{\text{given}} ACH_M$ is not regular $\Rightarrow \sim ACH_M$ is not regular

$\Rightarrow L(G)$ is not regular $\Rightarrow f(\langle M\rangle) \notin REG_{CFG}$.

$\therefore \langle M\rangle \in E_{TM} \Longleftrightarrow L(G) \in REG_{CFG}$

so $E_{TM} \leq_m REG_{CFG}$ and since E_{TM} is not decidable (or even r.e.), REG_{CFG} isn't either.

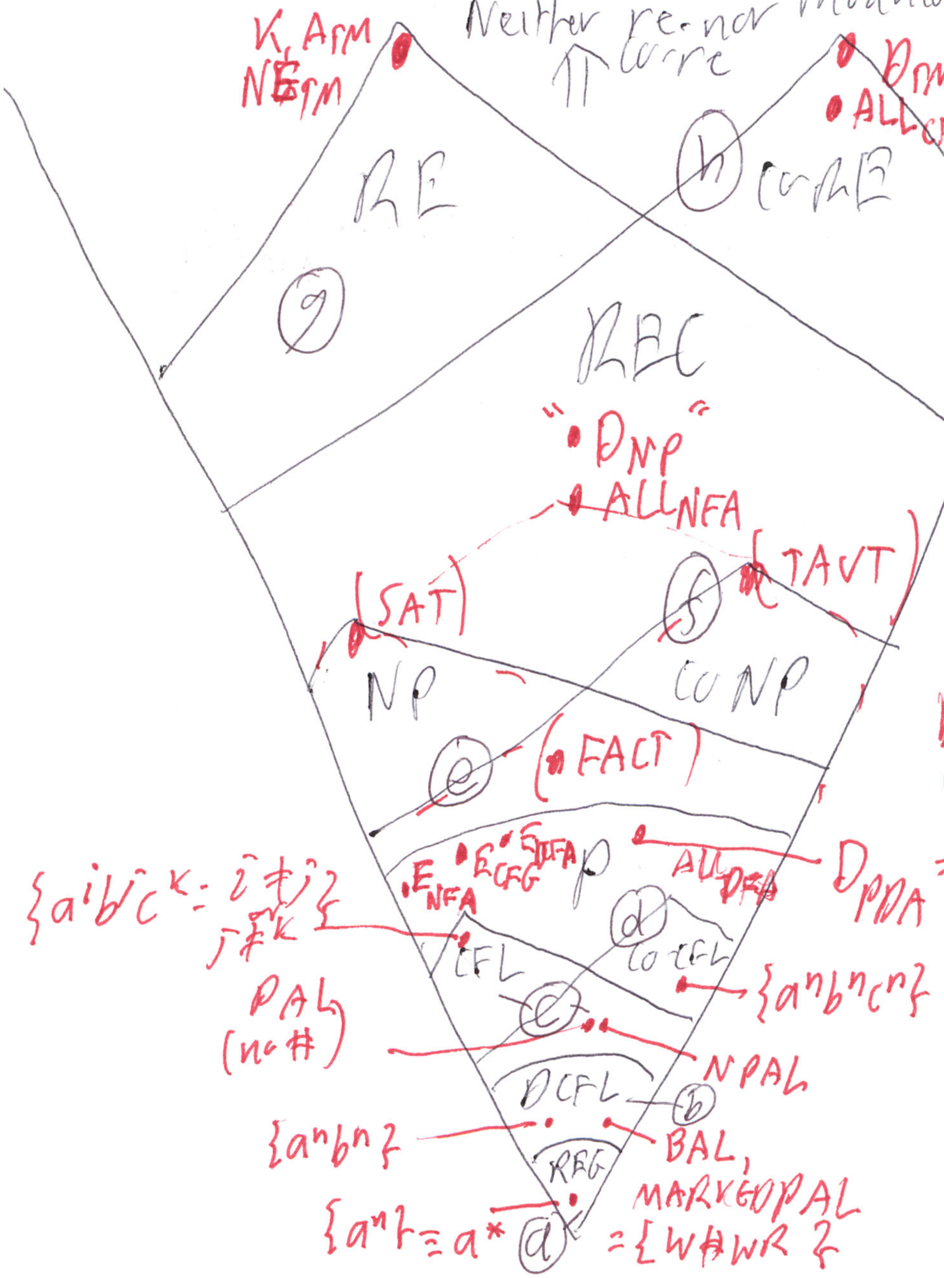
If we were able to show $A_{TM} \leq REG_{CFG}$
this would show the language REG_{CFG}
to be neither r.e. nor co-r.e.

Ex: $L = \{x \in \{a,b\}^* : x$ is ~~not~~ a palindrome$\}$ is a CFL.

$G: S \to aSa \mid bSb \mid \overbrace{aTb \mid bTa}$ "I screwed up"

$L(G) = NPAL$ $\quad T \to aT \mid bT \mid \varepsilon$

$\sim ACH_M \approx (a+b)^*$ "NPAL • ALL_{TM} " modulo δ_M $(a+b)^*$



A_{NFA} is in P ~~not~~ because we convert NFA to DFA but because we can trace the computation in quadratic time.

$D_{PDA} = \{P : P$ is a PDA that does not accept $\langle P \rangle\}$ "in" is in P because A_{CFG} is in P.

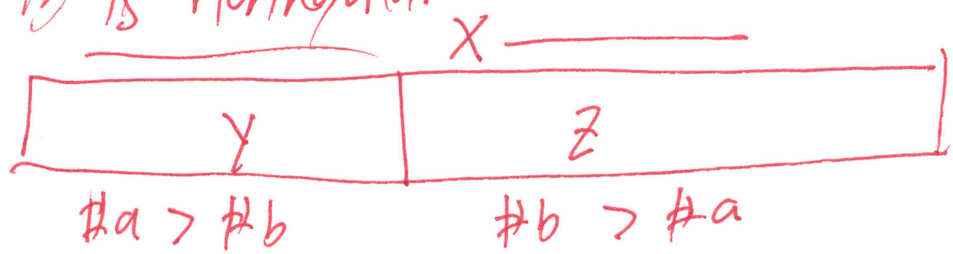
(not in text!)

(5') $A = \{x : \#a(x) > \#b(x)\}$

$B = \{x : \#a(x) < \#b(x)\}$

Show that $L = A \cdot B$ is nonregular

$x \in A \cdot B$ means



$x = a\underbrace{bbbbbb\ldots b}_{n}\underbrace{aaaaaa}_{n-1 \text{ a's}}$

- only have one possible breakdown.
- Add one more a then $xy \notin L$

Take $S = ab^+$ clearly S is infinite

Let any $x, y \in S$, $x \neq y$ be given. Then

$x = ab^m$, $y = ab^n$ where wlog $m < n$. $n-1 \geq m$

Take $z = a^{n-1}$. Then $yz = ab^n a^{n-1} \in L$ but $xz = ab^m a^{n-1} \notin L$

$\therefore L(xz) \neq L(yz)$. $\therefore L$ is nonregular. because it has too many trailing a's.

Guessing a string y st. ... is like an unbounded loop

```
n=0
while (true)
  for x = λ to 1^n
    try "x ∈ AEG (G)"
      if true, break and accept
  }
  n++
}
```