CSE 596 Lecture Fri 9/28 Fall 2018.

Say $x \in \{a,b\}^*$ has a "balancing b" if there is a b s.t. $x = ubv$ and $\#a(u) = \#a(v)$.

aaba no aababa yes

Prove the language L of such strings is not regular.

Take $S = \underline{a^*b}$ (Clearly S is infinite)

Let any $x, y \in S$, $x \neq y$ be given. Then we can write $x = \underline{a^m b}$ $y = \underline{a^n b}$ where $\underline{m \neq n}$ (could say $m < n$) (without loss of gen...)

Take $z = \underline{a^m}$

Then $xz \in L$ because $\underline{xz = a^m b a^m}$ b balances but $yz \notin L$ because $\underline{yz = a^n b a^m}$ and there's only one b which doesn't balance.

Thus $L(xz) \neq L(yz)$, and since x, y in S are arbitrary, S is an infinite PD set for L, so L is non-regular by MN...

If $(\exists S_{\text{infinite}})(\forall x, y \in S, x \neq y)(\exists z)\, L(xz) \neq L(yz)$, then L is not regular.

For Problem 2, consider cases like $x = 01$, $y = 011$. $z = x^R = 10$ doesn't work because $yz = 01110$ is a palindrome.

To prove that a K-state DFA M is minimum, give a PD set S of size K for the language.

Picking up on consequences of K-tapes-to-1 and Universal RAM-TM.

Consequence (4)

$M_i \equiv$ the TM whose instructions and components are coded by i as string or a Gödel number

Text

$\langle M \rangle$ stands for the string i when $M = M_i$.

We can assume i ends with ASCII NUL (\0) not otherwise used in the code, so we can parse $\langle M \rangle x$.

$A_{TM} = \{\underbrace{\langle M \rangle}_{\text{binary } i}\underbrace{x}_{\in\{0,1\}^* \text{ if } \Sigma=\{0,1\}}: M$ is a single-tape TM that accepts the input $x \in \Sigma^*\}$

Thus we can consider A_{TM} as a language $\subseteq \{0,1\}^*$. Or allow ASCII^ (etc.).

4. **Theorem**: There is a single-tape TM M_U s.t. not only is $L(M_U) = A_{TM}$, but M_U simulates $M(x)$ in an overt manner.

overt, i.e. obvious

Proof: The Turing Kit is a Java pgm that takes any M, x as input and executes $M(x)$.

$\therefore$ We can build a 3-tape TM M_p s.t. M_p simulates $TK(\underbrace{M,x}_{})$. Then convert M_p to 1-tape M_U. IS Literally $\langle M \rangle x$ on the tape of M_p and M_U.

M_U is a (pretty efficient!) Universal Turing Machine

5. For every NTM N we can build a DTM M st. $L(M) = L(N)$.

Proof: Using high-level programming we can d

(for t=1; ; t++) {

try all t-step possible computations of N on the given input X.

if any acceptance is found, accept X (by M).

}.

$X \in L(N) \Rightarrow N$ has a t-step accepting computation for some t.

$\Rightarrow M$ eventually finds it, so $X \in L(M)$.

$X \notin L(N) \Rightarrow M$ never accepts (and may never halt).

Convert this code to a DTM M.

Thus $L(M) = L(N)$, but even when $X \in L(N)$, $M(X)$ may run exponentially slower. $\boxtimes$

Added: The "for t=1; ; t++)" style loop is a useful trick to design not-necessarily halting programs to accept other c.e. languages.

"recognizable" in notes