Say $x \in \{a,b\}^*$ has a "balancing $b" if there is a $b$ st. $X = ubv$ and $|a|_w = |a|_v$.

$aaba \; \text{no} \; aababa \; \text{yes}$

Prove the language $L$ of such strings is not regular.

Take $S = \overline{a^*b}$ (clearly $S$ is infinite).

Let any $x, y \in S$, $x \neq y$ be given. Then we can write $x = a^m b$ and $y = a^n b$ where $m \neq n$.

Take $z = a^m$ without loss of gen.

Then $xz \in L$ because $b$ balances but

$yz \notin L$ because $y_2 z = a^m b a^m$ and there's only one $b$ which doesn't balance.

Thus $S$ is an infinite PO set for $L$, so $L$ is non-regular by MN

If ($\exists S$ 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 $x = 01$, $y = 011$. $z = x^2 = 10$ doesn't work.

The $z = 01110$ is a palindrome.

To prove that a $K$-state DFA $M$ is minimum, give a PO set $S$ of size $K$ for the language.
Picking up on consequences of K-tape-to-1 and Universal RAM-TM.

Mi = the TM whose instructions and components are coded by i as string or a Gödel number. \(\langle M \rangle\) stands for the string i when \(M = M_i\).

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

\[ \text{Atm} = \{ \langle M \rangle x : M \text{ is a single-tape TM that accepts the input } x \in \Sigma^* \}. \]

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

4. Theorem: There is a single-tape TM \(M_\emptyset\) s.t. not only \(L(M_\emptyset) = \text{Atm}\), but \(M_\emptyset\) simulates \(M(x)\) in an overt manner.

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

\(\therefore\) We can build a 3-tape TM \(M_\emptyset\) s.t. \(M_\emptyset\) simulates \(TK(M_i, x)\). Then convert \(M_\emptyset\) to 1-tape \(M_\emptyset\). Literally \(\langle M \rangle x\) on the tape of \(M_\emptyset\) and \(M_\emptyset\).

\(M_\emptyset\) is a (path efficient!) Universal Turing Machine.
5. For every NFA $N$ we can build a DFA $M$ s.t. $L(M) = L(N)$.

Proof: Using high-level programming, we can add:

```c
for (t = 1; ; t++) {
    try all t-step possible computations of $N$ on the given input $x$. (by M)
    if any acceptance is found, accept $x$.
}
```

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

$\implies M$ eventually finds it, so $x \in L(M)$.

$x \notin L(N) \implies M$ 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.

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