

Today's Q: What kind of formalism can most efficiently represent or decide these languages.

$BAL = \{x \in \{(,)\}^* : x \text{ is balanced}\}$ For all $i, 0 \leq i \leq n=|x|$, $\text{diff}(x,i) \geq 0$,
 when $\text{diff}(x,i) = \# \text{lefts} - \# \text{rights in } x_1 \dots x_i$.
 And $\text{diff}(x,n) = 0$ for final balance.

$PAL = \{x \in \{0,1\}^* : x \text{ is a palindrome}\}$ ie. $x = x^R$ x^R is x written backwards.

In of k , $L_k =$ the language of the regexp $(0+1)^* 1 (0+1)^{k-1}$

A more efficient recognizer for L_k is a "DFA with L moves allowed."

$B =$ "blank" after the end of x

Regarding L

- If I had to
- The NFA
- Fact: The

Take $S =$
 there is an
 Take $Z =$
 Then xz
 but yz
 and since $x,$
 can build a

I forgot to take a photo before erasing the lower half. What was originally there was a proof that PAL is non-regular using the Myhill-Nerode script. Here it is: Take $S = 0^*1$. Clearly S is infinite. Let any $x, y \in S$, $x \neq y$, be given. Then there are different natural numbers m and n such that $x = 0^m 1$ and $y = 0^n 1$. Take $z = 0^m$. Then $xz = 0^m 1 0^m$ which is a palindrome, but $yz = 0^n 1 0^m$ is not because $n \neq m$ and the 1 in the middle leaves no way of making it one. Thus $PAL(xz) \neq PAL(yz)$, so S is PD for PAL, and since S is infinite, PAL is non-regular by MNT.

I also had a side note that while you can also say "wlog. $m < n$ " this proof does not need it.

Regarding L_k , note:

- The "regexp with powering" $(0+1)^* \cdot 1 \cdot (0+1)^n$ has $O(\log k)$ chars.
- If I had to write $(0+1)^* \cdot 1 \cdot (0+1)(0+1) \dots (0+1)$ it would be $\Theta(k)$ chars.
- The NFA has $k+1$ states, so $\Theta(k)$ specification size, maybe $\Theta(k \log k)$.

$K-1$ steps, so K states

Fact: The smallest DFA M_k st. $L(M_k) = L_k$ has 2^k states. Proved by MNT.

Take $S = \{0,1\}^k$. Let any $x, y \in S$, $x \neq y$ be given. Then, numbering from 0, this time, there is an i st. x and y differ in bit i .

Take $z = 0^i$. Why suppose x has the string that has a '1' in place i , y the other.

Then xz has the '1' in position k from the end, so $xz \in L_k$, but yz has the '0' in the same position, so $yz \notin L_k$. Thus $L_k(xz) \neq L_k(yz)$, and since $x, y \in S$ are arbitrary, S is a PD set of size 2^k for L_k . And, we can build a DFA with 2^k states by tracking the previous k chars read in $0^k x$.

Today's Q: What kind of formalism can most efficiently represent or decide these languages:

$BAL = \{x \in \{ (,) \}^* : x \text{ is balanced}\}$ For all $i, 0 \leq i \leq n=|x|$, $diff(x,i) \geq 0$,
 when $diff(x,i) = diff \# \text{lefts} - \# \text{rights in } x_1 \dots x_i$.
 And $diff(x,n) = 0$ for final balance.

$PAL = \{x \in \{0,1\}^* : x \text{ is a palindrome}\}$ ie. $x = x^R$ x^R is x written backwards.

As a fn of k , $L_k =$ the language of the regexp $(0+1)^k 1 (0+1)^k$

A more efficient recognizer for L_k is a "DFA with L moves allowed."

Does this idea work for recognizing palindromes?

No. But if we can also X-out chars then we can.

To finish the design of this Turing Machine, we need to add termination logic. But we can see it runs in $\Theta(n^2)$ time.

Regarding L_k

- If I had to write $(0+1)^k \cdot 1 \cdot (0+1)^k$ it would be $\Theta(k)$ chars.
- The NFA has $k+1$ states, so $\Theta(k)$ specification size, maybe $\Theta(k \log k)$.

Fact: The smallest DFA M_k st. $L(M_k) = L_k$ has 2^k states. Prod by MNT.

Take $S = \{0,1\}^k$. Let any $x, y \in S, x \neq y$ be given. Then, numbering from 0 this time, there is an i st. x and y differ in bit i .

Take $z = 0^i$. Wlog suppose " x " names the string that has a "1" in place i , " y " the other.

Do a final streamed match of all chars in x and x^R .

we can do it in $O(n)$ time.

This is after sketching (most of!) a Turing Machine for PAL on the rest of the left-hand board. On the right-hand board I only added a sketch of a more-efficient two-tape TM on the bottom:

Regarding L_k , note:

- The "regexp with powering" $(0+1)^k \cdot 1 \cdot (0+1)^k$ has $O(\log k)$ chars.
- If I had to write $(0+1)^k \cdot 1 \cdot (0+1)^k$ it would be $\Theta(k)$ chars.
- The NFA has $k+1$ states, so $\Theta(k)$ specification size, maybe $\Theta(k \log k)$.

Fact: The smallest DFA M_k st. $L(M_k) = L_k$ has 2^k states. Prod by MNT.

Take $S = \{0,1\}^k$. Let any $x, y \in S, x \neq y$ be given. Then, numbering from 0 this time, there is an i st. x and y differ in bit i .

Take $z = 0^i$. Wlog suppose " x " names the string that has a "1" in place i , " y " the other.

Do a final streamed match of all chars in x and x^R .

we can do it in $O(n)$ time.

The missing bottom line says that with a second tape one can do it (that is, recognize the language PAL) in $O(n)$ time. Wednesday's lecture will define Turing machines formally and also the idea of running time.