

(1) FA to Regexp Example: Dragons & Spears DFA

Exit without holding a spear

$L_{SS} = \alpha + \$ ( \alpha + \$ )^* D$

$L_{S1} = L_{SS} \$ ( \alpha + \$ )^*$   
exit with one spear.

$L(M) = L_{SS} \cup L_{S1}$ . The dead state (can never be part of processing that accepts). So we can delete it. Then we have a 2-state base case:

$L_{SS} = (\alpha + \beta \gamma^*)^*$

$L_{S1} = L_{SS} \beta \gamma^* = \alpha \cdot \beta \cdot L_{SS}$

Why not  $L_{S1} = L_{SS} \alpha^* \$ ( \alpha + \$ )^*$ ?  
Is equally correct, but redundant because  $L_{SS}$  already includes a trailing  $\alpha$ .

How about  $M_2$ ?

Has 2 accepting states hence not so simple for the algorithm. Can add f, then remove states 1, 2 from F, eliminate them, and get the base case.

$L = \alpha^* (\epsilon + \beta)$

How about when we can save 2 spears? Can we "sightread" this DFA  $M_2$ ?

$L_{11}$  is "nice" because there no direct route between 1 and 2.

$L_{11} = ( \alpha + D \alpha^* \$ + \$ ( \alpha + \$ )^* D )^*$

Ways to go from state 1 back to 1 once.

$L(M) = L_{SS} \cup L_{S1} \cup L_{S2}$

$= ( \alpha + \$ \cdot L_{11} \cdot D )^* \cup \alpha^* \$ L_{11} \cup \alpha^* \$ \cdot L_{11} \cdot \$ \cdot ( \alpha + \$ )^*$

May feel redundant because the use of  $L_{11}$  means the inside is not "one". But it is correct.

Correct? Certainly sound. Is it comprehensive? Needed the final  $( \alpha + \$ )^*$ .

$\$ ( \alpha + \$ )^*$ ? base case

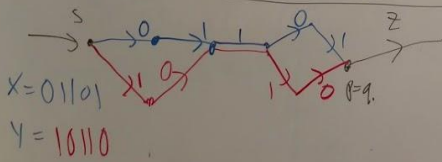
redundant & includes a trailing  $\alpha$

Definition: Given any language  $L$  (not necessarily regular), two strings  $x, y \in \Sigma^*$  are distinguishable for  $L$ , written  $x \not\sim_L y$ ,

if there is a string  $z \in \Sigma^*$  s.t.  $L(xz) \neq L(yz)$ .

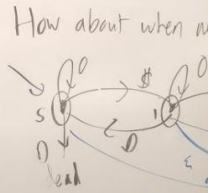
Negation is  $(\forall z \in \Sigma^*) L(xz) = L(yz)$ , then write  $x \sim_L y$ .

Key Point: If  $x \not\sim_L y$ , then any DFA  $M$  such that  $L(M) = L$  must process  $x, y$  from  $s$  to different states p, q.



Upon starting any suffix  $z$ , the processing would stay converged, so  $xz$  and  $yz$  would end at the same state  $r$ . But  $r$  cannot be both acc and rej.

Then  $M$  could not be correct for  $L$ .

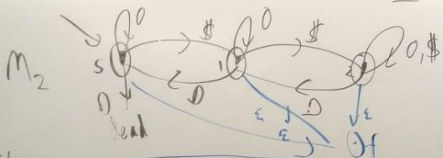


How about when  $w$ ...  
Eg.  $w = \epsilon$   
Now suppose we have  $w, x, y$  s.t.  $w \not\sim_L x$   
Then we need 3 d... to process them to

Def<sup>n</sup>: A set  $S \subseteq \Sigma^*$  is pairwise distinctive for  $L$  if for all distinct  $x, y \in S$ ,  $x \not\sim_L y$ .

Key Lemma: If  $\|S\| = k$ , s.t.  $L(M) = L$  must have

How about when we can save 2 spaces?



with the  $\begin{cases} 1 & \text{if } x \in L \\ 0 & \text{otherwise} \end{cases}$   
 $yz \in L$   
 $(xz \in L \wedge yz \in L)$

Eg.  $w = \epsilon, x = \$, y = \$\$$ .

Now suppose we have 3 strings  $w, x, y$  s.t.  $w \not\sim_L x, w \not\sim_L y, x \not\sim_L y$ .

Then we need 3 different states to process them to.

Then  $M$  could not be correct for  $L$ .

Def<sup>n</sup>: A set  $S \subseteq \Sigma^*$  is pairwise distinctive for  $L$  if for all distinct pairs  $x, y \in S$ ,  $x \not\sim_L y$ .

Key Lemma: If  $\|S\| = k$ , then any DFA  $M$  s.t.  $L(M) = L$  must have at least  $k$  states.

John T 1987, VB Anil still alive, at Cornell

Myhill-Nerode Theorem not nec SSL.

(a) If a language  $L$  has an infinite PD set  $S$ , then  $L$  is not regular.

(b) Moreover, every nonregular lang. has an infinite PD set.

Proof for (a): Read the "Key Lemma" with  $k = \infty$ .

Every  $M$  would need  $\infty$ -many states, but then  $M$  would not be a finite automaton. So  $L$  has no DFA, so  $L$  is not regular.

Note also that  $\{\epsilon, \$, \$\$, D\}$  is a PD set of size 4, so  $M_2$  is optimal.