

Outline

- Finish off Regex \rightarrow e-NFA \rightarrow NFA \rightarrow DFA \rightarrow Regex
- Minimization/equivalence (Myhill-Nerode theorem)



Summary

- NFA \rightarrow DFA
 - *If NFA has states Q , construct a DFA with 2^Q states*
- E-NFA \rightarrow DFA
 - *Change the transition relation to include epsilon-edges*
- Regex \rightarrow e-NFA
 - *Construct e-NFA for each regex by structural induction*
- DFA \rightarrow regex
 - *That's next*



DFA \rightarrow Regex

Theorem If L is accepted by a DFA, then L is denoted by a regular expression.

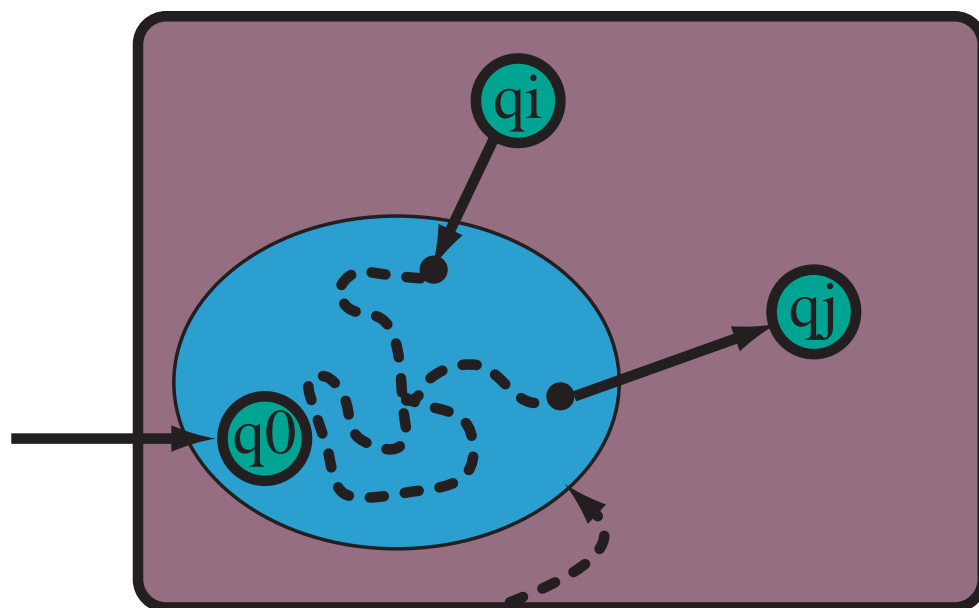
Proof Outline:

- Consider some arbitrary DFA $M = (Q, \Sigma, \delta, s, F)$
- Clearly, we want to do induction over *something*
 - Choices could be $|Q|$, what else?



DFA \rightarrow Regex

- Let $Q = \{q_1, q_2, \dots, q_n\}$
- We'll do induction over k for executions that use only states in $\{q_1, q_2, \dots, q_k\}$ (this is non-obvious)



States numbered $\leq k$



Defining R^k_{ij}

- Let R^k_{ij} be the set of all strings x such that $\delta(q_i, x) = q_j$ and for any proper prefix y of x that is not ϵ , $\delta(q_i, y) = q_l \wedge l \leq k$
- By induction:
 - $R^0_{ij} = \begin{cases} \{a \mid \delta(q_i, a) = q_j\} & \text{if } i \neq j \\ \{a \mid \delta(q_i, a) = q_j\} \cup \{\epsilon\} & \text{if } i = j \end{cases}$
 - $R^k_{ij} = R^{k-1}_{ik} (R^{k-1}_{kk})^* R^{k-1}_{kj} \cup R^{k-1}_{ij}$
- Note that $L(M) = \bigcup_{f \in F} R^n_{0f}$



Constructing the regex

Theorem There is a regular expression r_{ij}^k that denotes each R_{ij}^k

Proof By induction on k

Note $L(M) = r_{0j_1}^n + \dots + r_{0j_m}^n$ for $F = \{a_{j_1}, \dots, a_{j_m}\}$



Constructing the regex

Base • Have $R_{ij}^0 = \begin{cases} \{a \mid \delta(q_i, a) = q_j\} & \text{if } i \neq j \\ \{a \mid \delta(q_i, a) = q_j\} \cup \{\epsilon\} & \text{if } i = j \end{cases}$

• Let $r_{ij}^0 = \begin{cases} a_1 + a_2 + \dots + a_p & \text{for } i \neq j \\ a_1 + a_2 + \dots + a_p + \epsilon & \text{for } i = j \end{cases}$

Step • Have $R_{ij}^k = R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1} \cup R_{ij}^{k-1}$

• Let $r_{ij}^k = r_{ik}^{k-1} (r_{kk}^{k-1})^* r_{kj}^{k-1} + r_{ij}^{k-1}$



Minimizing FA

- Can we prove equivalence of regexs (or their FA)?
- One method:
 - *If each FA can be reduced to a unique minimal form, we can compare the minimal FA*



Right-invariance

- Let $M = (Q, \Sigma, \delta, s, F)$ be a DFA
- For each $x, y \in \Sigma^*$, let $xR_M y \Leftrightarrow \delta(s, x) = \delta(s, y)$
- R_M is an equivalence relation:

Reflexive $xR_M x$

Symmetric If $xR_M y$ then $yR_M x$

Transitive If $xR_M y$ and $yR_M z$, then $xR_M z$

- Furthermore if $xR_M y$, then $xzR_M yz$ for any $z \in \Sigma^*$



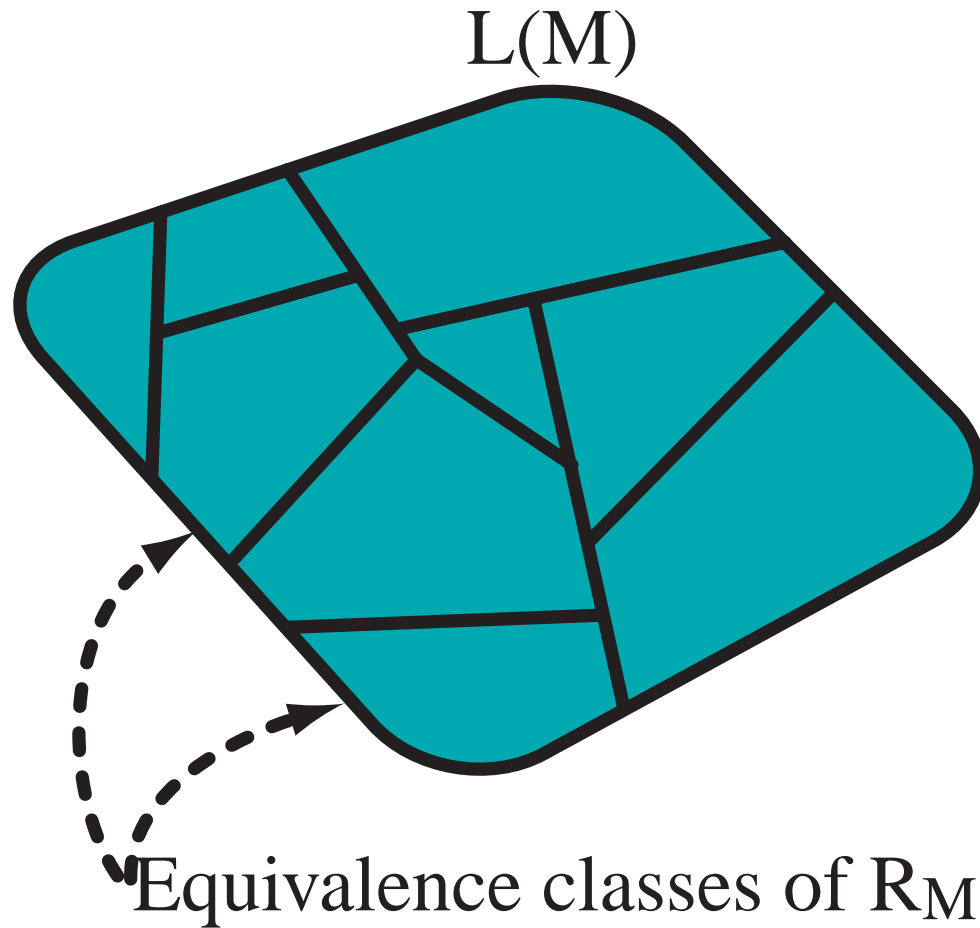
Right-invariance

- Furthermore if $xR_M y$, then $xzR_M yz$ for any $z \in \Sigma^*$

Definition For any relation R , if xRy implies $xzRyz$, then R is *right invariant* with respect to concatenation.



Equivalence classes of R_M



Myhill-Nerode

Theorem (The Myhill-Nerode theorem). The following three statements are equivalent:

- The set $L \subseteq \Sigma^*$ is accepted by a FA.
- The set L is the union of some of the equivalence classes of a right-invariant equivalence relation with finite index.
- Let equivalence relation R_L be defined as xR_Ly iff $xz \in L \Leftrightarrow yz \in L$. Then R_L has finite index.



1->2

- Let M be a DFA that accepts L .
- Define R_M as before.
- R_M is right invariant
- R_M has finite index because the DFA has only a finite number of states
- L is the union of the equivalence classes containing some X where $\delta(s, x) \in F$



2->3

- Consider some equivalence relation E of finite index, and L is the union of some of its equivalence classes.
- Prove R_L has finite index, where xR_Ly iff $xz \in L \Leftrightarrow yz \in L$.
 - Suppose xEy
 - Since E is right-invariant, then $xzEyz$
 - Then $xz \in L \Leftrightarrow yz \in L$, and so xR_Ly
 - Since $xEy \Rightarrow xR_Ly$, and E has finite index, so does R_L



3->1 (part 1)

- First, prove R_L is right-invariant.
 - Suppose xR_Ly ; prove xwR_Lyw
 - This means, prove $xwzR_Lywz$ for any $z \in \Sigma^*$
 - But xR_Ly implies $xv \in L \Leftrightarrow yv \in L$
 - If $v = wz$, then $xwz \in L \Leftrightarrow ywz \in L$
 - So xwR_Lyw

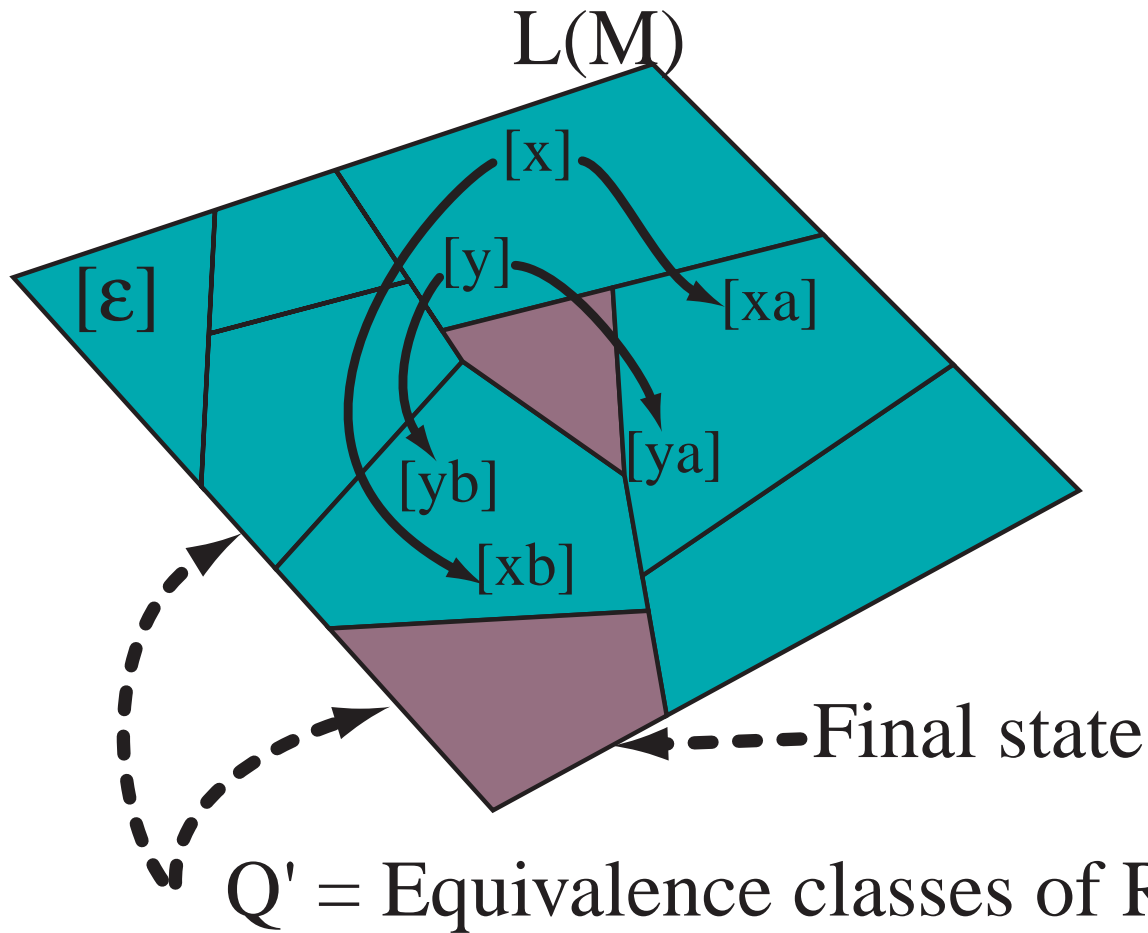


3->1 (part 2)

- Construct DFA $M' = (Q', \Sigma, \delta', s', F')$ from R_L
 - Let Q' be the set of equivalence classes of L (there are a finite number)
 - Define $[x] \in Q'$ to be the equivalence class containing x
 - Let $\delta'([x], a) = [xa]$
 - * Is $\delta'([x], a)$ a function?
 - * Yes, because R_L is right invariant
 - Let $s' = [\epsilon]$
 - Let $F' = \{[x] \mid x \in L\}$



3- \rightarrow 1 DFA construction



Minimization

- The minimum automaton accepting a language L is unique (up to renaming of states), and is given by the construction in the Myhill-Nerode theorem.



Minimization

- The minimum automaton accepting a language L is unique (up to renaming of states), and is given by the construction in the Myhill-Nerode theorem.
- An *algorithm* is a procedure that always terminates (with the right answer)



Minimization algorithm

- Let $M = (Q, \Sigma, \delta, s, F)$ be a DFA.
- Define $p \equiv q$ iff for each x , $\delta(p, x) \in F \Leftrightarrow \delta(q, x) \in F$.
- Basic idea:
 - $p \not\equiv q$ if $p \in F$ and $q \notin F$
 - Consider $r = \delta(p, a)$ and $s = \delta(q, a)$
 - If $r \not\equiv s$, then $p \not\equiv q$
 - Compute the fixpoint



Pseudo-code

```
let  $D = \{(p, q) \mid p \in F \wedge q \notin F\}$ 
do
  for  $(p, q) \in F \times F \cup (Q - F) \times (Q - F)$  do
    if  $\delta(p, a), \delta(q, a) \in D$  for some  $a$  then
       $D \leftarrow D \cup (p, q)$ 
until  $D$  doesn't change any more
```



Finishing up

- To build the minimal DFA: let $[q] = \{p \mid (p, q) \notin D\}$
- Define $\delta([q], a) = [\delta(q, a)]$
- Remove inaccessible states (states not reachable from the start state)



Minimization proof

- Let $M = (Q, \Sigma, \delta, s, F)$ be the original DFA,
- Let $M = (Q', \Sigma, \delta', s', F')$ be the constructed DFA.
 - $Q' = \{[q] \mid q \text{ is accessible from } s\}$
 - $\delta'([q], a) = [\delta'(q, a)]$
 - $s' = [s]$
 - $F' = \{[q] \mid q \in F\}$



Minimization proof

- First, δ' is a function
- Second, use induction to show $\delta'([s], w) = [\delta(s, w)]$, so $L(M) = L(M')$
- Is M' minimal?
 - Suppose not; then there are state $[p] \neq [q]$ where $\delta(s, x) = p$ and $\delta(s, y) = q$ and $xR_L y$.
 - There *must* be some w that distinguishes p and q (according to the algorithm)
 - If so, $xwR_L yw$ is false, which cannot be.



Summary

- FAs/regex are equivalent
- For each FA there is a unique minimal FA'
 - *Equivalence of FA is computable*
 - *To test if $FA1=FA2$, minimize both and test for equality*
 - *Remember that the states may have different names, so this may take some time*
- Next: what languages are regular?



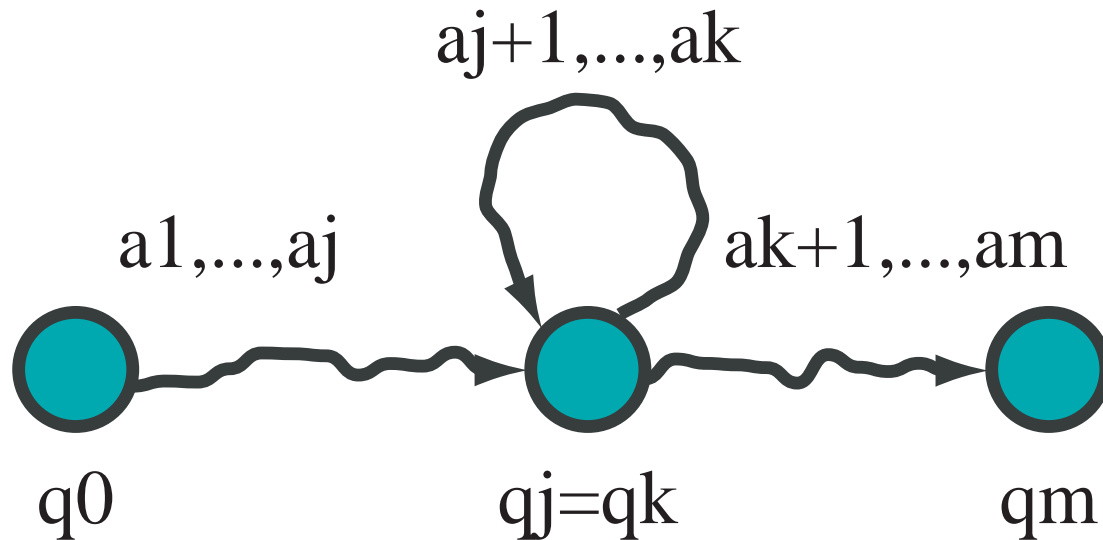
What languages are not regular?

- Intuition: since a FA has only finite state, it can “remember” only a finite number of things
- Some things we would expect are not regular
 - *Balanced parentheses (have to remember an arbitrary nesting depth)*
 - *Prime numbers*
 - *Why are multiples of 3 regular?*
 - *Addition (strings “ $i+j=k$ ”)*



Regular languages

- Intuition: if a FA accepts a string that is “long enough,” it must repeat a state
 - *But it can't remember that the state was repeated*
 - *So it can be forced to repeat the state over and over*



Pumping lemma

Lemma (the Pumping Lemma)

- Let L be a regular set.
- There is a constant n s.t. for any z where $|z| \geq n$, then z can be written $z = uvw$, where
 - $|uv| \leq n$
 - $|v| \geq 1$
 - For all $i \geq 0$, $uv^i w \in L$
 - n is bounded by $|Q|$



Proving that a language is not regular

- Let L be the proposed regular language
- There is some n , by the pumping lemma
- Choose a string s , longer than n symbols, in the language L
- Using the pumping lemma, construct a new string s' that is not in the language



Balanced parentheses

The strings of balanced parentheses are not regular.

Proof

- Let n be the integer in the pumping lemma
- Consider the string $s = (^{n+1})^{n+1}$
- Write $s = uvw$ where $uv = (^n$ and $w =)^{n+1}$
- Then $s' = uv^2w = (^{n+m+1})^{n+1}$ is in the language for some $m > 0$
- s' is not balanced



Unions

- Any finite union of regular sets is regular
- Infinite unions may not be regular



Executions

- Consider a NFA $M = (Q, \Sigma, \delta, s, F)$
- The executions $(q_0, c_0)(q_1, c_1) \dots (q_{n-1}, c_{n-1})q_n$ are regular

