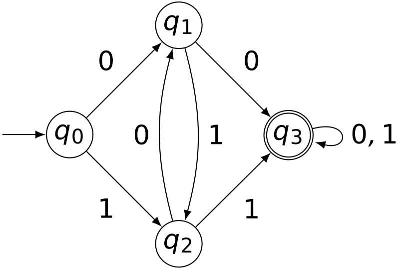
DFSA → RE

Intuition

- Let $A = (Q, \Sigma, \delta, s, F)$ be a DFA
- Create a RE R by eliminating states and replacing transition labels by more complex REs
  - For each accepting state $q \in F$, eliminate all states except s and q to get RE R_q
  - Overall RE for A is the union ("+") of R_q for all $q \in F$

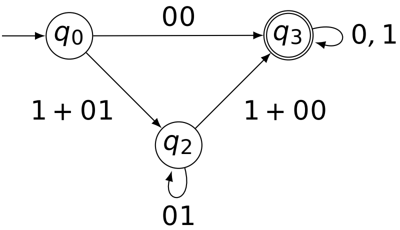
Example 2.24 *(cont'd)*

$A =$ 

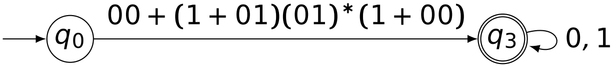
Example 2.24 *(cont'd)*

Construction of RE for $\mathcal{L}(A)$

- First, eliminate q_1 (we could have started with q_2):



- Next, eliminate q_2:

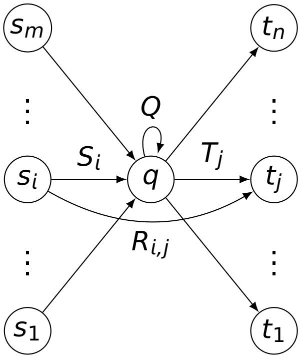


- $R_{q_3} = \big(00 + (1 + 01)(01)^*(1 + 00)\big)(0 + 1)^*$
- Since q_3 is the only accepting state of A, $R = R_{q_3}$ satisfies $\mathcal{L}(R) = \mathcal{L}(A)$

DFSA → RE *(cont'd)*

In general

- To eliminate state q, consider **all** states that have a transition into or from q
  - Note: possible for same state to appear on both sides (as s_k and t_ℓ)
- For every pair s_i, t_j,
  - let S_i be the RE labelling the transition $s_i \to q$
  - let T_j be the RE labelling the transition $q \to t_j$
  - let Q be the RE labelling the loop on state q ($Q = \varnothing$ if there is no such loop)
  - let $R_{i,j}$ be the RE labelling the transition $s_i \to t_j$ ($R_{i,j} = \varnothing$ if there is no such transition)
- After the elimination of state q:
  - Label $R_{i,j}$ is replaced by $R_{i,j} + S_i Q^* T_j$
  - if $R_{i,j} = \varnothing$, this adds new transition $s_i \xrightarrow{S_i Q^* T_j} t_j$
  - if $Q = \varnothing$, "$S_i Q^* T_j$" simplifies to $S_i T_j$



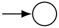
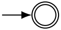
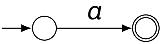
DFSA $\rightarrow$ RE *(cont'd)*

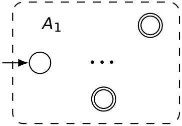
Putting it all together

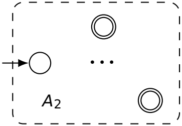
- For accepting state $q \in F$,
  - eliminate every state except q and initial state s
  - *this includes other accepting states*
  - if $q = s$ is the initial state, eliminate every state except q
  - result is a RE R_q
- If $F = \{q_1, q_2, \ldots, q_k\}$
  - do this separately for each accepting state to obtain REs $R_{q_1}, R_{q_2}, \ldots, R_{q_k}$
  - start with the entire DFA each time
  - final result $R_A = R_{q_1} + R_{q_2} + \cdots + R_{q_k}$
  - claim: $\mathcal{L}(R_A) = \mathcal{L}(A)$

RE $\longrightarrow$ NFSA

In general

- Define NFA based on recursive structure of RE
- Basic REs:
  - For $\varnothing$: 
  - For ε: 
  - For all $a \in \Sigma$: 

- Recursive REs:
  - For all REs R_1, R_2, let
    - A_1 = 

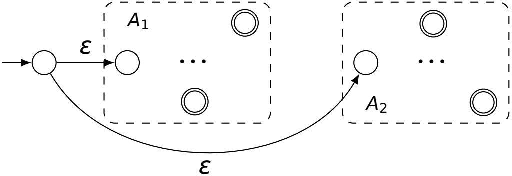
(could have any number of accepting states)
    - A_2 = 

(could have any number of accepting states)

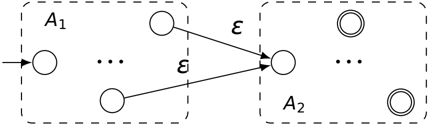
    be NFA such that $\mathcal{L}(A_1) = \mathcal{L}(R_1)$, $\mathcal{L}(A_2) = \mathcal{L}(R_2)$

RE $\longrightarrow$ NFSA *(cont'd)*

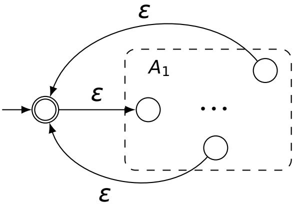
- Recursive constructions *(cont'd)*:
  - NFA for $R_1 + R_2$:



  - NFA for $R_1 \cdot R_2$:



  - NFA for R_1^*:



Closure properties for Regular languages

What is "closure"?

- The property of a set to be closed under certain operations
- In this case what *language* operations can we apply to regular languages, to create new languages that are still regular?
- Set operations (union, intersection, complement)? Yes
- Concatenation, Kleene star, exponentiation (for **fixed** exponents)? Yes

Example 2.25

- Suppose $L \subseteq \{0, 1, 2\}^*$ is regular, show that L' is also regular, where

$$L' = \Big\{ w \in \{0,1,2\}^* : w = 1x \text{ for some } x \in L, \text{ or} \\ w = 0x \text{ for some } x \in \overline{L} \Big\}$$

- Can be argued based on FSA (i.e., DFA or NFA) or based on REs

Example 2.25 *(cont'd)*

FSA-based argument

- By definition, there exists a DFA $A_1 = (Q_1, \{0, 1, 2\}, \delta_1, s_1, F_1)$ such that $\mathcal{L}(A_1) = L$
- Since L is regular, so is $\overline{L}$, so there also exists a DFA $A_2 = (Q_2, \{0, 1, 2\}, \delta_2, s_2, F_2)$ such that $\mathcal{L}(A_2) = \overline{L}$
- Create a DFA $A = (Q, \{0, 1, 2\}, \delta, q_0, F)$ as follows:
  - $Q = \{q_0\} \cup Q_1 \cup Q_2$
    (relabel states in Q_1 and Q_2 if necessary, so $q_0 \notin Q_1 \cup Q_2$ and $Q_1 \cap Q_2 = \varnothing$)
  - $\delta(q_0, 1) = s_1$, $\delta(q_0, 0) = s_2$, $\delta(q, a) = \delta_1(q, a)$ for all $q \in Q_1, a \in \{0, 1, 2\}$,
    $\delta(q, a) = \delta_2(q, a)$ for all $q \in Q_2, a \in \{0, 1, 2\}$
  - $F = F_1 \cup F_2$
- Then, A accepts w iff
  - $w = 1x$ for some string x accepted by A_1, or
  - $w = 0x$ for some string x accepted by A_2
- In other words, $\mathcal{L}(A) = L'$

Example 2.25 *(cont'd)*

RE-based argument

- By definition, there exists a RE R_1 over $\{0, 1, 2\}$ such that $\mathcal{L}(R_1) = L$
- Since L is regular, so is $\overline{L}$, so there also exists a RE R_2 such that $\mathcal{L}(R_2) = \overline{L}$
- Then, $R' = 1R_1 + 0R_2$ is also a RE over $\{0, 1, 2\}$ and

$$\begin{aligned}\mathcal{L}(R'') &= \mathcal{L}(1R_1) \cup \mathcal{L}(0R_2) \\ &= (1 \cdot \mathcal{L}(R_1)) \cup (0 \cdot \mathcal{L}(R_2)) \\ &= (1 \cdot L) \cup (0 \cdot \overline{L}) \\ &= L'\end{aligned}$$

Does this mean **every** language is regular?...

Example 2.26: Is $L_0 = \{a^n b^n : n \in \mathbb{N}\}$ regular?

Explore

- Q: Why does $\mathcal{L}(a^* b^*) \neq L_0$?
  A: $a^* b^*$ allows non-equal numbers of a's and b's
- Issue: using a fixed number of states to keep track of an arbitrary number of a's
- Conjecture: L_0 is not regular
- How do we prove this?
- Idea: prove no DFA recognizes exactly L_0, i.e., for all DFA A, $\mathcal{L}(A) \neq L_0$
- In other words, every DFA "makes a mistake": either some string in L_0 is rejected by the DFA, or some string not in L_0 is accepted by the DFA
- Logically equivalent to proving every DFA that accepts all strings in L_0 also accepts some string not in L_0

Example 2.26 *(cont'd)*

Proof

- Let $A = (Q, \Sigma, \delta, s, F)$ be an arbitrary DFA
- Assume A accepts every string in L_0 ($L_0 \subseteq \mathcal{L}(A)$)
  - Note: assumption says nothing about the strings rejected by A
- Let $K = |Q|$ (number of states of A)
- Consider computation of A on input $w = a^{K+1}b^{K+1}$
- Because w contains more a's than number of states in A, computation must go through at least one state twice while processing only a's
- In other words, there is some state $q \in Q$ and some $1 \leqslant i \leqslant K$ such that $\delta^*(q, a^i) = q$
- When A is in state q, it has no way to know how many times it has been through this state before
- Since A accepts w (by assumption), A must also accept $w' = a^{K+1+i}b^{K+1}$ by repeating the loop from state q back to itself one more time
- Since $i \geqslant 1$, $a^{K+1+i}b^{K+1} \notin L_0$, so $\mathcal{L}(A) \neq L_0$ $\square$

Observations on Regular Languages

In general

- If $L \subseteq \Sigma^*$ is regular, then $L = \mathcal{L}(A)$ for some DFA A
  - A accepts every $w \in L$
  - A rejects every $w \in \overline{L} = \Sigma^* - L$
  - Note: A is fixed—the same A makes the correct decision for every input w
- let p be the number of states of A
- let $w \in L$ be a string with $|w| > p$
  - Q: does such a string always exist?
  - A: not necessarily: L could be finite
- What happens when A processes w?
  - at least one state is repeated at least once
  - in other words, the path taken by A contains a loop
  - let x be the part of w processed before the loop, y the part processed during the loop, and z the part processed after the loop
  - then A accepts not only $w = xyz$, but also xz (do not take the loop), xy^2z (take the loop twice), xy^3z (take the loop three times), ...

The Pumping Lemma

Theorem 2.27: Pumping Lemma

- *For all regular languages $L \subseteq \Sigma^*$,*
- *there exists a positive integer $p \in \mathbb{Z}^+$ such that*
- *for every string $w \in L$ with $|w| \geqslant p$*
- *there are sub-strings $x, y, z \in \Sigma^*$ such that*
  - *$w = xyz$,*
  - *$|xy| \leqslant p$,*
  - *$|y| \geqslant 1$, and*
  - *$xy^iz \in L$ for all $i \in \mathbb{N}$.*

Discussion

- Preceding slide contains high-level proof idea
- Imposes a condition on the structure of all “long enough” strings in L
- What if L does not contain such strings?
  - Can only happen if L is finite
  - Lemma is still true for p larger than the length of a longest string in L

How does this help?

Pumping Lemma: applies to **all** regular languages

- For all regular languages $L \subseteq \Sigma^*$,
- there exists $p \in \mathbb{Z}^+$ such that
- for every string $w \in L$ with $|w| \geqslant p$
- there are $x, y, z \in \Sigma^*$ such that
  - $w = xyz$ and
  - $|xy| \leqslant p$ and
  - $|y| \geqslant 1$ and
- $xy^iz \in L$ for all $i \in \mathbb{N}$.

Application: proving language L non-regular (by contradiction)

- Assume $L \subseteq \Sigma^*$ is regular
- let $p \in \mathbb{Z}^+$ (arbitrary)
- choose $w \in L$ with $|w| \geqslant p$
- let $x, y, z \in \Sigma^*$ (arbitrary) such that
  - $w = xyz$ and
  - $|xy| \leqslant p$ and
  - $|y| \geqslant 1$, then
- choose $i \in \mathbb{N}$ such that $xy^iz \notin L$.

Example 2.28: $L_0 = \{0^n 1^n : n \in \mathbb{N}\}$

Proof

- Assume L_0 is regular
- Let $p \in \mathbb{Z}^+$
- Choose $w = 0^p 1^p$
- Let $x, y, z \in \{0, 1\}^*$ such that
  - $w = xyz$,
  - $|xy| \leqslant p$,
  - $|y| \geqslant 1$
- Then xy consists entirely of 0's so $y = 0^k$ for some $k > 0$
- This implies $xy^2z = 0^{p+k}1^p \notin L_0$ $\square$