Lecture 3

Thursday, February 16, 2023 11:00 AM

1.3. Regular Expressions, Nondeterminism, and Kleene's Theorem (Chapter 3)

- Reg Expression, Nondet, and Kleene
- Reg Lang and Reg expressions
  - Many simple lang can be expressed by formula involving languages w/ single string of length 1 and ops of union, concatenation and Kleene star
    - Is called the kleene star
  - **Examples:**
    - Strings ending in *aa*: $\{a, b\}^* \{aa\}$
      (This is a simplification of $(\{a\} \cup \{b\})^*\{a\}\{a\}$).
    - Strings containing *ab* or *bba*. What is the regular expression for that (get a *)? $\{a, b\}^* \{ab, bba\} \{a, b\}^*$.
    - The language $\{a, b\}^* \{aa, aab\}^* \{b\}$. What strings are included? (get a *) ■

    All these are called *regular* languages.

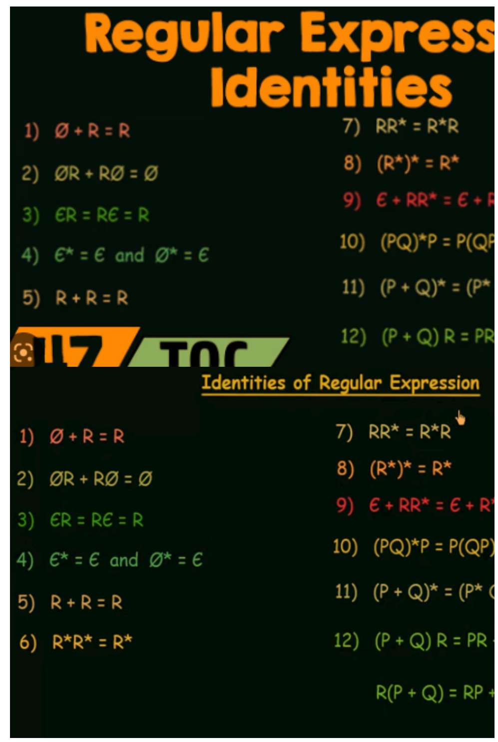
    So it can start with anything, then either have the pattern aa or aab any number of times, then ends with b
  -
  - **Definition:** If Σ is an alphabet, the set $\boldsymbol{R}$ of regular languages over Σ is defined as follows:
    - The language $\varnothing$ is an element of $\boldsymbol{R}$, and for every $\sigma \in \Sigma$, the language $\{\sigma\}$ is in $\boldsymbol{R}$.
    - For every two languages L_1 and L_2 in $\boldsymbol{R}$, the three languages $L_1 \cup L_2$, L_1L_2, and $L_1{}^*$ are elements of $\boldsymbol{R}$. ■

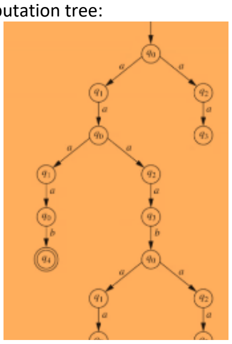
    Examples:
    - $\{\Lambda\}$, because $\varnothing^* = \{\Lambda\}$.
    - $\{a, b\}^*\{aa\} = (\{a\} \cup \{b\})^* (\{a\}\{a\})$. ■
  -
  -
  -
  - Language that is null is an element of R and for every sigma an element of Sigma, the language {sigma} is in R
  - {lambda} include bc null* = {lambda}
  -
  - Reg expression for lang is little more user friendly
    - Parentheses replace curly braces (used only when needed)
    - Union symbol replaced by +
    -

Regular language	*Regular Expression*
$\varnothing$	$\varnothing$
$\{\Lambda\}$	Λ
$\{a,b\}^*$	$(a+b)^*$
$\{aab\}^*\{a,ab\}$	$(aab)^*(a+ab)$

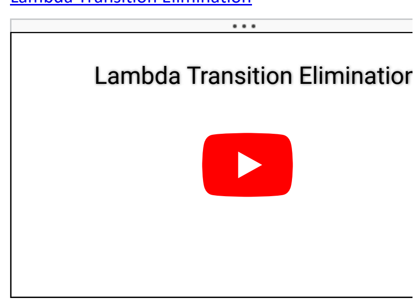
  - Table of converting reg lang to reg expression (linux grep)
  - 2 reg expressions are equal if the languages they describe are equal
  - See if reg expressions are equal
    - **Examples:**
      - Does $(a^*b^*)^*$ equal $(a+b)^*$? (get a *) ■
      - Does $(a+b)^*ab(a+b)^*+b^*a^*$ equal $(a+b)^*$? (get a *) ■

      i. Yes, they are equal
      ii. Yes, this is fun

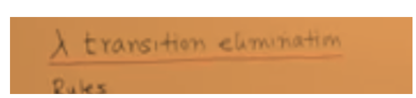


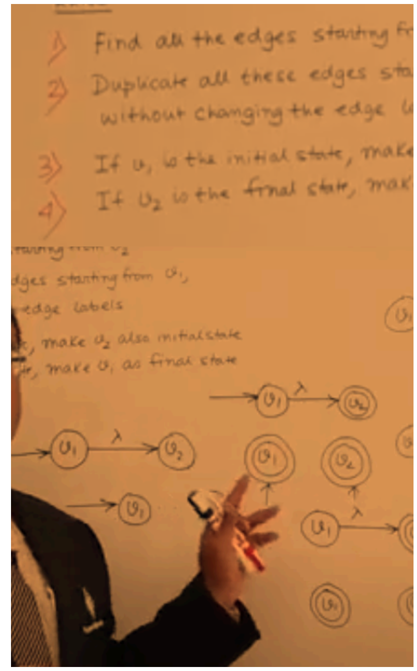
- NFA's
  - FA are now DFA, bc others are nondeterministic (NFA)
  - NFA's don't have to have both a and b exiting,
  - NFA don't describe algorithm for recognizing lang, but rather number of diff sequences of steps that can be followed
  - Computation tree:
    - 

[Lambda Transition Elimination](https://www.youtube.com/watch?v=ZyjqR9JYHgc)



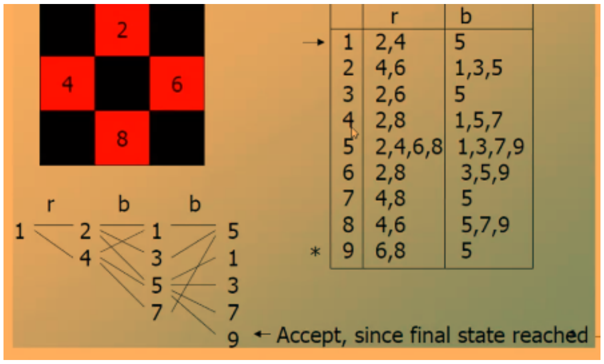
https://www.youtube.com/watch?v=ZyjqR9JYHgc



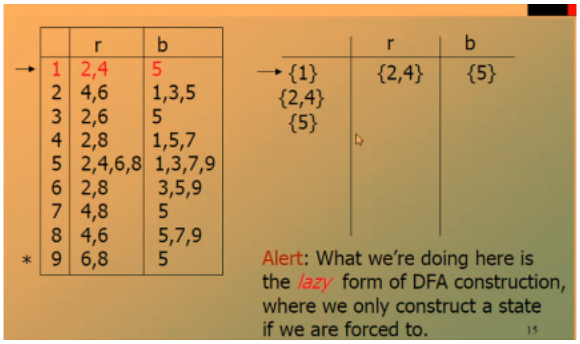


- Each lvl goes to 1 prefix of the string
- Less stars are better, 1 star is first lvl, need less loops
-
- NFA Def:
- **Definition:** A *nondeterministic finite automaton* (NFA) is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$, where:
  - Q is a finite set of states,
  - Σ is a finite input alphabet,
  - $q_0 \in Q$ is the initial state,
  - $A \subseteq Q$ is the set of accepting states,
  - $\delta : Q \times (\Sigma \cup \{\Lambda\}) \to 2^Q$ is the transition function. ■
  - **Note:** The values of δ are not single states, but *sets* of states.
- Can jump from state to state without letter, can change mind when jumping
- Multiple Lambda transitions can be on one state
  - Delta star for nfa is que
  - Lambda is I have an input, I don't want to use it, jump around without anything
  - Depending on how many lambda transitions u have, can just jump around without even touching input
- For any finite set defined recursively, easy form an algorithm to calculate lamda(s) -- looks like A(S) but without middle bar of the A
- **Definition:** Suppose $M = (Q, \Sigma, q_0, A, \delta)$ is an NFA and $S \subseteq Q$ is a set of states.
  - The Λ-*closure* of S is the set $\Lambda(S)$ that can be defined recursively as follows:
    - $S \subseteq \Lambda(S)$.
    - For every $q \in \Lambda(S)$, $\delta(q, \Lambda) \subseteq \Lambda(S)$.
- **Definition:** Let $M=(Q, \Sigma, q_0, A, \delta)$ be an NFA.
  - Define the extended transition function $\delta^* : Q \times \Sigma^* \to 2^Q$ as follows:
    - For every $q \in Q$, $\delta^*(q,\Lambda) = \Lambda(\{q\})$.
    - For every $q \in Q$, every $y \in \Sigma^*$, and every $\sigma \in \Sigma$:
      - $\delta^*(q, y\sigma) = \Lambda(\cup \{\delta(p, \sigma) \mid p \in \delta^*(q, y)\})$.
    - A string $x \in \Sigma^*$ is accepted by M if $\delta^*(q_0, x) \cap A \neq \varnothing$. (i.e., some sequence of transitions involving the symbols of x and Λ's leads from q_0 to an accepting state).
  - The language $L(M)$ accepted by M is the set of all strings accepted by M. ■

- Nondeterminism in a NFA can be eliminated
  - 2 diff types of nondeterminism:
    - Diff arcs for the same input symbol
    - and lambda transitions
  - For lamda transitions, intro new transitions so that we don't need lambda transitions
    - When there isn't delta transitions from p to q, but the NFA can go from p to q using 1/more lambda transitions as well as delta, we intro delta transitions
    - Resulting nfa can have more nondeterminism of the first type, but no lambda transitions
    - "fluffy automata"
  -
  - **Theorem:** For every language $L \subseteq \Sigma^*$ accepted by an NFA $M = (Q, \Sigma, q_0, A, \delta)$, there is an NFA M_1 with no Λ-transitions that also accepts L. ■
    - **Proof.** Define $M_1 = (Q, \Sigma, q_0, A_1, \delta_1)$, where for every $q \in Q$, $\delta_1(q, \Lambda) = \varnothing$, and for every $q \in Q$ and every $\sigma \in \Sigma$, $\delta_1(q, \sigma) = \delta^*(q, \sigma)$.
  - So I can do lambda transitions at the beginning, and then I can do lambda transition again
  - Define $A_1 = A \cup \{q_0\}$ if $\Lambda \in L$, and $A_1 = A$ otherwise.
    - We can prove, by structural induction on x, that for every q and every x with $|x| \geq 1$, $\delta_1{}^*(q, x) = \delta^*(q, x)$. ■
  -
  -
  - Slide 49
  - Algorithm by Andrei that gets min numb of states
  - Chess board- red and black transitions (black squares have 1, 3, 5, 7, 9), its non-deterministic, can use palindrome
    - "Que of sets"
    -

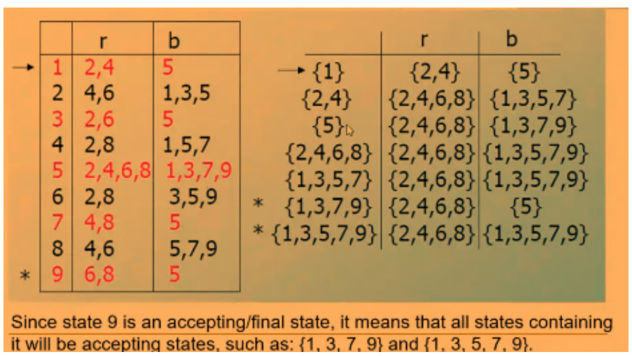
- 

		r	b
→	1	2,4	5
	2	4,6	1,3,5
	3	2,6	5
	4	2,8	1,5,7
	5	2,4,6,8	1,3,7,9
	6	2,8	3,5,9
	7	4,8	5
	8	4,6	5,7,9
*	9	6,8	5

- If go like palindrome, go from front and back to find shortest path, meet in middle (5)
- 

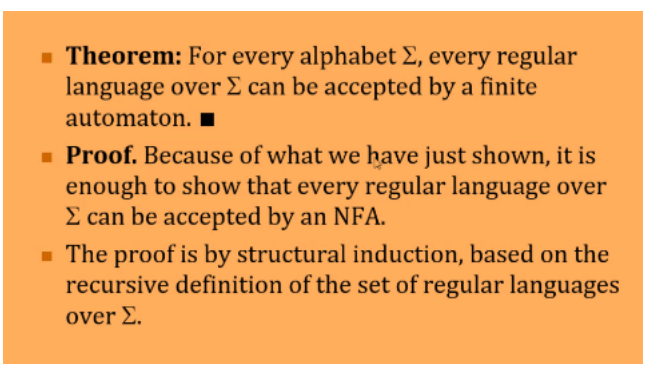
		r	b
→	1	2,4	5
	2	4,6	1,3,5
	3	2,6	5
	4	2,8	1,5,7
	5	2,4,6,8	1,3,7,9
	6	2,8	3,5,9
	7	4,8	5
	8	4,6	5,7,9
*	9	6,8	5

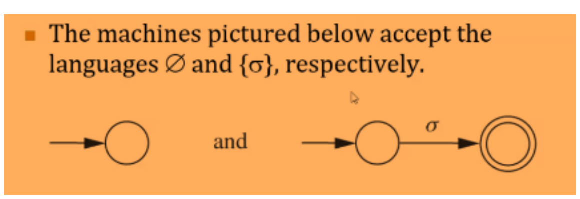
		r	b
→	{1}	{2,4}	{5}
	{2,4}		
	{5}		

- 

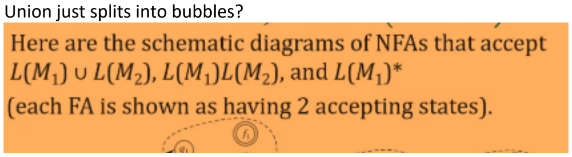
		r	b
→	1	2,4	5
	2	4,6	1,3,5
	3	2,6	5
	4	2,8	1,5,7
	5	2,4,6,8	1,3,7,9
	6	2,8	3,5,9
	7	4,8	5
	8	4,6	5,7,9
*	9	6,8	5

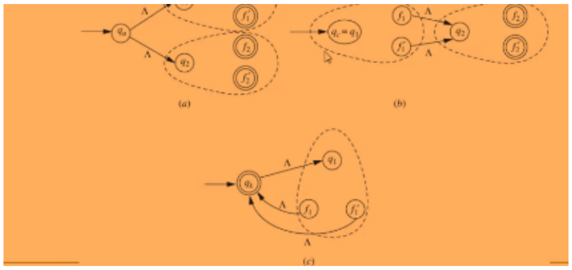
		r	b
→	{1}	{2,4}	{5}
	{2,4}	{2,4,6,8}	{1,3,5,7}
	{5}	{2,4,6,8}	{1,3,7,9}
	{2,4,6,8}	{2,4,6,8}	{1,3,5,7,9}
	{1,3,5,7}	{2,4,6,8}	{1,3,5,7,9}
*	{1,3,7,9}	{2,4,6,8}	{5}
*	{1,3,5,7,9}	{2,4,6,8}	{1,3,5,7,9}

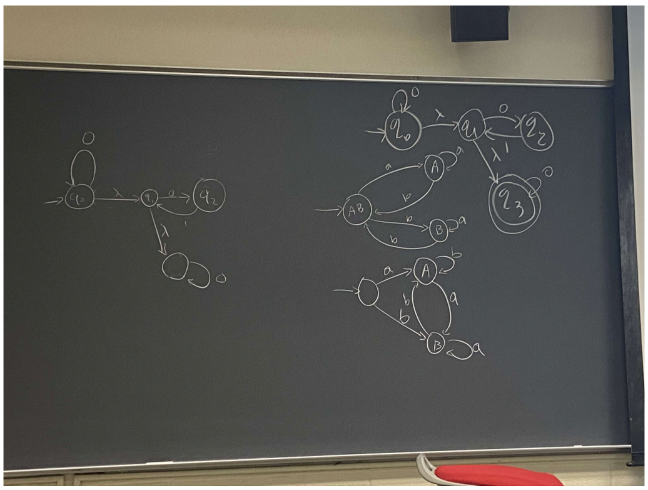
- 9 for NFA, 7 states for DFA
- NFA to DFA
- Kleene's Theory
- 

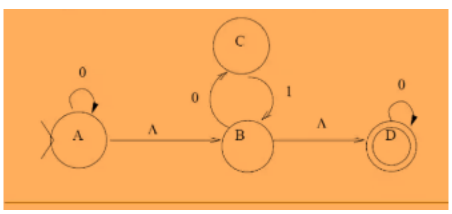
- 

- Base cases are easy
- Union just splits into bubbles?

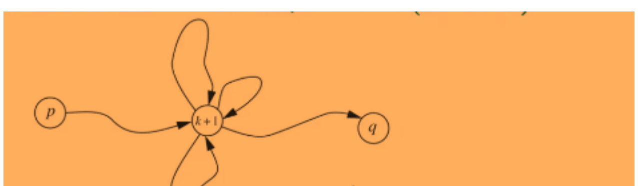


- 
- All accepting states have to have transition state to second transition
- By using these constructions, we can create for every regular expression an NFA that accepts the corresponding language. ■
- (get a star) What is an NFA for accepting the regular expression $0^*(01)^*0^*$?





- **Theorem:** For every finite automaton $M=(Q, \Sigma, q_0, A, \delta)$, the language $L(M)$ is regular. ■
- **Proof:** First, for two states p and q, we define the language $L(p, q) = \{x \in \Sigma^* \mid \delta^*(p, x)=q\}$.
- If we can show that for every p and q in Q, $L(p, q)$ is regular, then it will follow that $L(M)$ is, because
  - $L(M) = \cup\{L(q_0, q) \mid q \in A\}$.
  - The union of a finite collection of regular languages is regular.
- We will show that $L(p, q)$ is regular by expressing it in terms of simpler languages that are regular.



- Every string in $L(p, q, k+1)$ can be described in one of those two ways and every string that has one of these two forms is in $L(p, q, k+1)$. This leads to the formula
  - $L(p, q, k+1) = L(p, q, k) \cup$

$$L(p, k{+}1, k)\ L(k{+}1, k{+}1, k)^*\ L(k{+}1, q, k)$$

- Slides 67
- Ex, lang is (aa+aab)*b