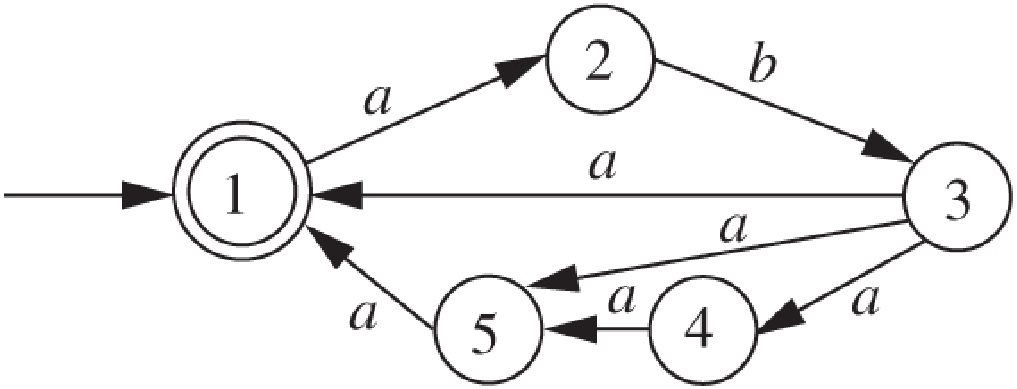
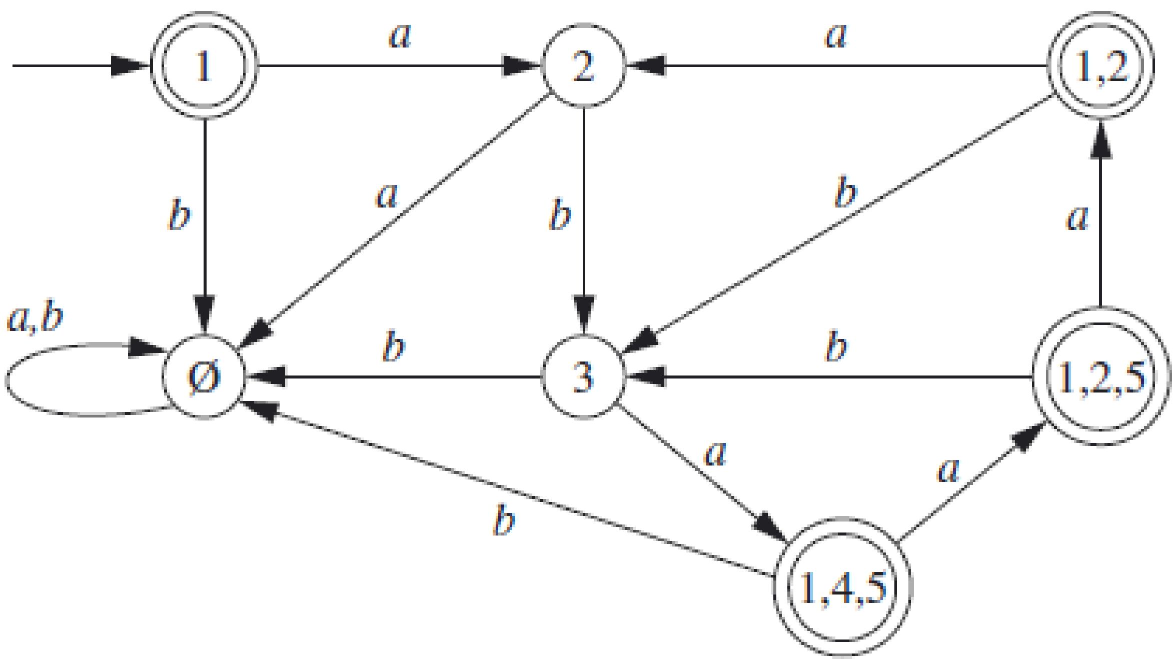


q	δ(q,a)	δ(q,b)
1	{2}	∅
2	∅	{3}
3	{1,4,5}	∅
4	{5}	∅
5	{1}	∅



Kleene's Theorem, Part 1

- **Theorem: For every alphabet Σ, every regular language over Σ can be accepted by a finite automaton**
- Because of what we have just shown, it is enough to show that every regular language over Σ can be accepted by an NFA
- The proof is by structural induction, based on the recursive definition of the set of regular languages over Σ

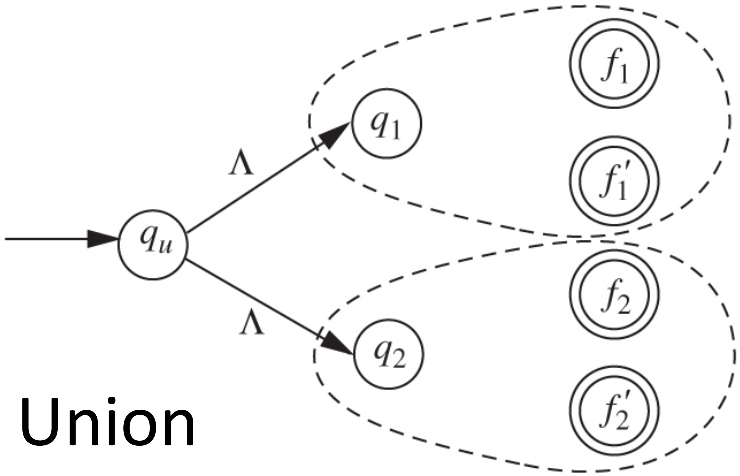
Homework: Learn both parts of Kleene's theorem (including proofs).

Kleene's Theorem, Part 1 (cont'd.)

- The basis cases are easy
- The automata pictured below accept the languages ∅ and {σ}, respectively

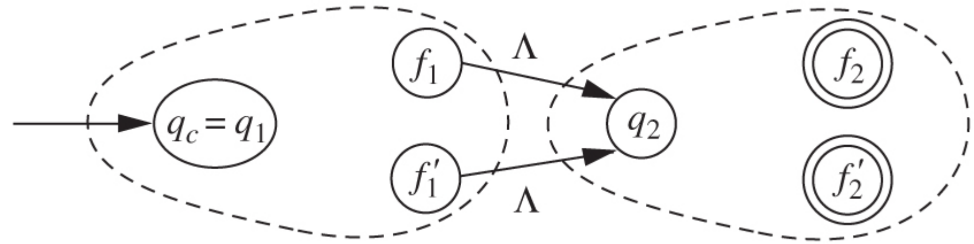
- Induction hypothesis: both L_1 and L_2 are regular languages can be accepted by NFAs
- Induction step: $L(M_1) \cup L(M_2)$, $L(M_1)L(M_2)$, and $L(M_1)^*$ can be accepted by NFAs

Each FA is shown as having 2 accepting states

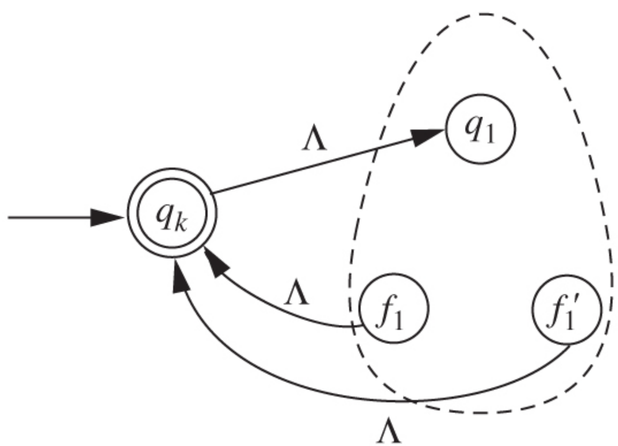


Union

(a)



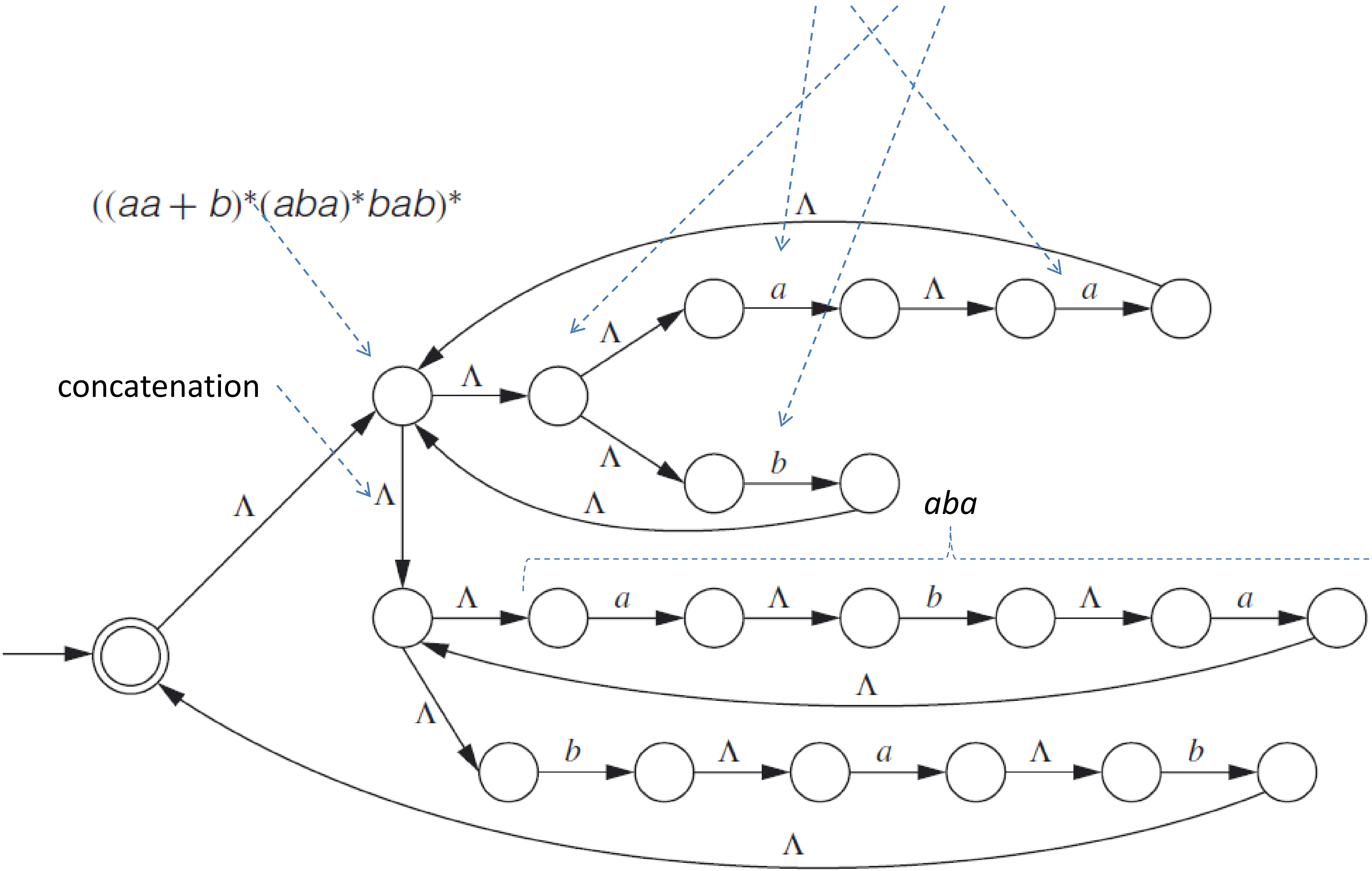
(b) Concatenation



Kleene's *

(c)

An NFA Corresponding to $((aa + b)^*(aba)^*bab)^*$



Kleene's Theorem, Part 2

- **Theorem: For every finite automaton $\boldsymbol{M}$=($\boldsymbol{Q}$, Σ, $\boldsymbol{q_0}$, $\boldsymbol{A}$, δ), the language $\boldsymbol{L}$($\boldsymbol{M}$) 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 …
  - $\boldsymbol{L(M) = \cup \{L(q_0, q) \mid q \in A\}}$
  - 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

Kleene’s Theorem, Part 2 (cont’d.)

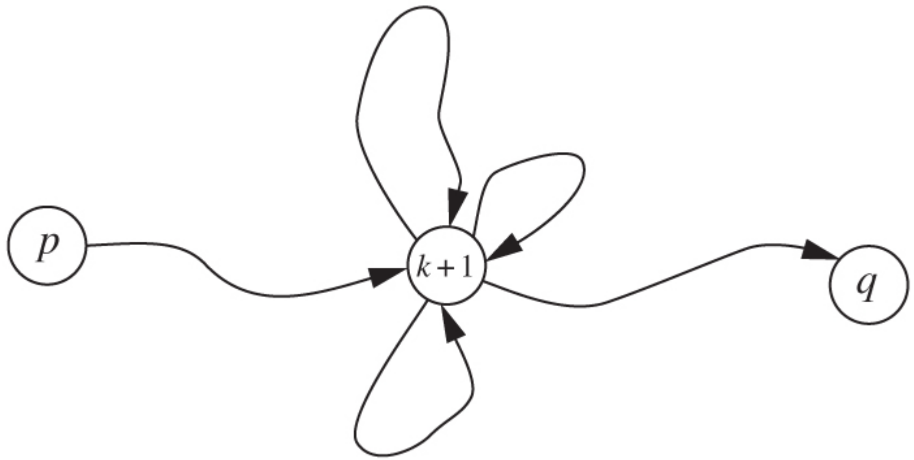
- We will consider the distinct states through which M passes as it moves from p to q
- If $x \in L(p, q)$, we say x causes M to go from p to q through a state r if there are non-null strings x_1 and x_2 such that $x = x_1x_2$, $\delta^*(p, x_1) = r$, and $\delta^*(r, x_2) = q$
  - In using a string of length 1 to go from p to q, M does not go *through* any state
  - How can we construct an inductive proof on what happens between p and q?

Kleene's Theorem, Part 2 (cont'd.)

- Assume Q has n elements numbered 1 to n
- For $p, q \in Q$ and $j \geq 0$

  $L(p, q, j)$ = strings in $L(p, q)$ that cause M to go from p to q without going through any state numbered higher than j
- Suppose that for some number $k \geq 0$, $L(p, q, k)$ is regular for every $p, q \in Q$ and consider how a string can be in $L(p, q, k+1)$
  - The easiest way is for it to be in $L(p, q, k)$
  - If not, it causes M to go to $k+1$ one or more times, but M goes through nothing higher (i.e., no state $k+2$ for example)

Kleene's Theorem, Part 2 (cont'd.)



- 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
  - $L(p, q, k+1) = L(p, q, k) \cup$
    $L(p, k+1, k)\ L(k+1, k+1, k)^*\ L(k+1, q, k)$
- This is the main point of a proof by induction on k and for an algorithm

Algorithm: FA→RE. Let $r(i,j,k)$ denote a RE for $L(i,j,k)$. Then $L(M)$ is described by the RE

we need accepting states only

$$r(M) = r(1,1,3) + r(1,2,3)$$

The recursive formula with smaller k in the proof tells

$$\begin{aligned} r(1,1,3) &= r(1,1,2) + r(1,3,2)r(3,3,2)^*r(3,1,2) \\ r(1,2,3) &= r(1,2,2) + r(1,3,2)r(3,3,2)^*r(3,2,2) \end{aligned}$$

Applying the formula to the expressions $r(i,j,2)$ we get

$$r(1,1,2) = r(1,1,1) + r(1,2,1)r(2,2,1)^*r(2,1,1)$$

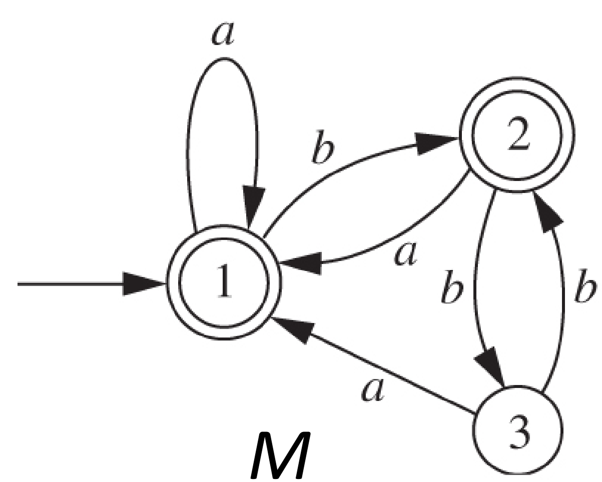
$$r(1,3,2) = r(1,3,1) + r(1,2,1)r(2,2,1)^*r(2,3,1)$$

$$r(3,3,2) = r(3,3,1) + r(3,2,1)r(2,2,1)^*r(2,3,1)$$

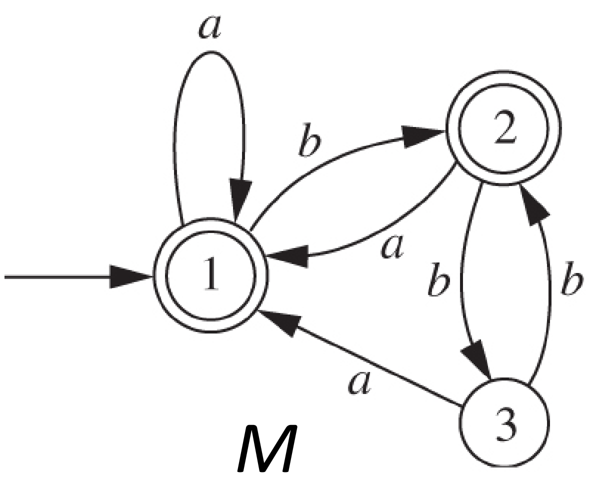
$$r(3,1,2) = r(3,1,1) + r(3,2,1)r(2,2,1)^*r(2,1,1)$$

$$r(1,2,2) = r(1,2,1) + r(1,2,1)r(2,2,1)^*r(2,2,1)$$

$$r(3,2,2) = r(3,2,1) + r(3,2,1)r(2,2,1)^*r(2,2,1)$$



p	$r(p,1,0)$	$r(p,2,0)$	$r(p,3,0)$
1	$a+\Lambda$	b	$\emptyset$
2	a	Λ	b
3	a	b	Λ



p	$r(p,1,1)$	$r(p,2,1)$	$r(p,3,1)$
1	a^*	a^*b	$\emptyset$
2	aa^*	$\Lambda + aa^*b$	b
3	aa^*	a^*b	Λ

p	$r(p,1,2)$	$r(p,2,2)$	$r(p,3,2)$
1	$a^*(baa^*)^*$	$a^*(baa^*)^*b$	$a^*(baa^*)^*bb$
2	$aa^*(baa^*)^*$	$(aa^*b)^*$	$(aa^*b)^*b$
3	$aa^* + a^*baa^*(baa^*)^*$	$a^*b(aa^*b)^*$	$\Lambda + a^*b(aa^*b)^*b$

Example:

$$\begin{aligned} r(2,2,1) &= r(2,2,0) + r(2,1,0)r(1,1,0)^*r(1,2,0) \\ &= \Lambda + (a)(a+\Lambda)^*(b) \\ &= \Lambda + aa^*b \end{aligned}$$

Regular languages

=

Languages of regular expressions

=

Languages accepted by FA

=

Languages accepted by NFA

Regular expressions and finite automata

- Tools such as grep, awk, and sed
- Email servers
- Pattern matching

Some motivation

Finite automata

- Software testing/QC
- TCP/IP, HTTP, and other protocols
- Hardware

Grammars, Automata, Regular Expressions

- GUI
- Lexical analysis in compilers of programming languages like C/C++, Java, and many more

Future computers

- Biomolecular finite automata
- DNA/RNA Turing machines

More questions

Some motivation

- Find duplicate occurrences of a phrase (Reg Exp).
- Does a program contain an assertion violation? Does a device driver respect certain protocols? (Properties of Lang)
- Can your software be stuck in an infinite loop? (Lang Incl)
- Does a distributed algorithm contain a livelock? (Lang Incl)
- Detect malicious Javascript entered into a web application. The set of malicious strings is a language. (Langs Inters)
- Run-time monitoring of reactive and mission-critical systems (nuclear reactors, chemical procs). (FA, Incl/Inters)
- Bioinformatics: pattern matching → build a language
- AI: FAs are used in simulation of character behavior