Regular Expressions = Regular Languages

Mark Greenstreet, CpSc 421, Term 1, 2006/07

Lecture Outline

Regular Expressions

- Regular Expresssions
- Equivalence of Regular Expressions and Finite Automata

Regular Madlibs

Once upon a $\underline{\hspace{6em}}_{\text{noun}}$, there was a $\underline{\hspace{6em}}_{\text{noun}}$ that $\underline{\hspace{10em}}_{\text{past tense verb}}$

$\underline{\hspace{14em}}_{\text{zero or more adjectives}}$ $\underline{\hspace{8em}}_{\text{plural noun}}$.

- Let `avocado` denote the language $\{$`avocado`$\}$.
- Let $noun =$ `avocado` $\cup$ `beach` $\cup$ `carrot` $\cup$ `caterpillar` $\cup$ `pencil` $\cup$ `penguins` $\cup$ `zombie`.
- Let $pluralNoun = noun$ `s`.
- Let $verb =$ `add` $\cup$ `compile` $\cup$ `eat` $\cup$ `sing` $\cup$ `swim` $\cup$ `walk`.
- Let $pastVerb = verb$ `ed`.
- Let $adjective =$ `beautiful` $\cup$ `big` $\cup$ `cold` $\cup$ `considerable` $\cup$ `furry` $\cup$ `insipid` $\cup$ `yellow`.
- Now, our Madlib$^{\text{TM}}$ is

  `Once upon a` *`noun`* `, there was a` *`noun`* `, that` *`pastVerb`* $($*`adjective`*$)^*$ *`pluralNoun.`*

Regular Madlibs

Once upon a ________ (noun), there was a ________ (noun) that ______________ (past tense verb) ______________________ (zero or more adjectives) ___________ (plural noun).

- Let `avocado` denote the language $\{$`avocado`$\}$.
- Let $noun =$ `avocado` $\cup$ `beach` $\cup$ `carrot` $\cup$ `caterpillar` $\cup$ `pencil` $\cup$ `penguins` $\cup$ `zombie`.
- Let $pluralNoun = noun$ `s`.
- Let $verb =$ `add` $\cup$ `compile` $\cup$ `eat` $\cup$ `sing` $\cup$ `swim` $\cup$ `walk`.
- Let $pastVerb = verb$ `ed`.
- Let $adjective =$ `beautiful` $\cup$ `big` $\cup$ `cold` $\cup$ `considerable` $\cup$ `furry` $\cup$ `insipid` $\cup$ `yellow`.
- Now, our Madlib™ is

  `Once upon a` `pencil` `, there was a` *`noun`* `, that` *`pastVerb`* $($*`adjective`*$)^*$ *`pluralNoun.`*

Regular Madlibs

Once upon a $\underset{\text{noun}}{\underline{\hspace{5em}}}$, there was a $\underset{\text{noun}}{\underline{\hspace{5em}}}$ that $\underset{\text{past tense verb}}{\underline{\hspace{8em}}}$

$\underset{\text{zero or more adjectives}}{\underline{\hspace{12em}}}$ $\underset{\text{plural noun}}{\underline{\hspace{6em}}}$.

- Let `avocado` denote the language $\{$`avocado`$\}$.
- Let $noun =$ `avocado` $\cup$ `beach` $\cup$ `carrot` $\cup$ `caterpillar` $\cup$ `pencil` $\cup$ `penguins` $\cup$ `zombie`.
- Let $pluralNoun = noun$ `s`.
- Let $verb =$ `add` $\cup$ `compile` $\cup$ `eat` $\cup$ `sing` $\cup$ `swim` $\cup$ `walk`.
- Let $pastVerb = verb$ `ed`.
- Let $adjective =$ `beautiful` $\cup$ `big` $\cup$ `cold` $\cup$ `considerable` $\cup$ `furry` $\cup$ `insipid` $\cup$ `yellow`.
- Now, our Madlib™ is

  `Once upon a` `pencil` `, there was a` `carrot` `, that`
  `pastVerb` $($*`adjective`*$)^*$ *`pluralNoun.`*

Regular Madlibs

Once upon a $\underline{\hspace{3em}}_{\text{noun}}$, there was a $\underline{\hspace{3em}}_{\text{noun}}$ that $\underline{\hspace{5em}}_{\text{past tense verb}}$ $\underline{\hspace{8em}}_{\text{zero or more adjectives}}$ $\underline{\hspace{4em}}_{\text{plural noun}}$.

- Let `avocado` denote the language $\{\texttt{avocado}\}$.
- Let $noun =$ `avocado` $\cup$ `beach` $\cup$ `carrot` $\cup$ `caterpillar` $\cup$ `pencil` $\cup$ `penguins` $\cup$ `zombie`.
- Let $pluralNoun = noun$ `s`.
- Let $verb =$ `add` $\cup$ `compile` $\cup$ `eat` $\cup$ `sing` $\cup$ `swim` $\cup$ `walk`.
- Let $pastVerb = verb$ `ed`.
- Let $adjective =$ `beautiful` $\cup$ `big` $\cup$ `cold` $\cup$ `considerable` $\cup$ `furry` $\cup$ `insipid` $\cup$ `yellow`.
- Now, our Madlib™ is

  `Once upon a pencil , there was a carrot , that walked` $(\textit{adjective})^*$ *pluralNoun*.

Regular Madlibs

Once upon a $\underset{\text{noun}}{\underline{\hspace{2cm}}}$, there was a $\underset{\text{noun}}{\underline{\hspace{2cm}}}$ that $\underset{\text{past tense verb}}{\underline{\hspace{3cm}}}$

$\underset{\text{zero or more adjectives}}{\underline{\hspace{5cm}}}$ $\underset{\text{plural noun}}{\underline{\hspace{2.5cm}}}$.

- Let `avocado` denote the language $\{$`avocado`$\}$.
- Let $noun =$
  `avocado` $\cup$ `beach` $\cup$ `carrot` $\cup$ `caterpillar` $\cup$ `pencil` $\cup$ `penguins` $\cup$ `zombie`.
- Let $pluralNoun = noun$ `s`.
- Let $verb =$ `add` $\cup$ `compile` $\cup$ `eat` $\cup$ `sing` $\cup$ `swim` $\cup$ `walk`.
- Let $pastVerb = verb$ `ed`.
- Let $adjective =$
  `beautiful` $\cup$ `big` $\cup$ `cold` $\cup$ `considerable` $\cup$ `furry` $\cup$ `insipid` $\cup$ `yellow`.
- Now, our Madlib™ is

  `Once upon a` `pencil` `, there was a` `carrot` `, that`
  `walked` `beautiful,` $($*`adjective`*$)^*$ *`pluralNoun.`*

Regular Madlibs

Once upon a $\underset{\text{noun}}{\underline{\hspace{5em}}}$, there was a $\underset{\text{noun}}{\underline{\hspace{5em}}}$ that $\underset{\text{past tense verb}}{\underline{\hspace{9em}}}$

$\underset{\text{zero or more adjectives}}{\underline{\hspace{13em}}}$ $\underset{\text{plural noun}}{\underline{\hspace{7em}}}$.

- Let `avocado` denote the language $\{$`avocado`$\}$.
- Let $noun =$ `avocado` $\cup$ `beach` $\cup$ `carrot` $\cup$ `caterpillar` $\cup$ `pencil` $\cup$ `penguins` $\cup$ `zombie`.
- Let $pluralNoun = noun$ `s`.
- Let $verb =$ `add` $\cup$ `compile` $\cup$ `eat` $\cup$ `sing` $\cup$ `swim` $\cup$ `walk`.
- Let $pastVerb = verb$ `ed`.
- Let $adjective =$ `beautiful` $\cup$ `big` $\cup$ `cold` $\cup$ `considerable` $\cup$ `furry` $\cup$ `insipid` $\cup$ `yellow`.
- Now, our Madlib$^{\text{TM}}$ is

```
Once upon a  pencil , there was a  carrot , that
walked   beautiful,  considerable   pluralNoun.
```

Regular Madlibs

Once upon a ________ (noun), there was a ________ (noun) that ______________ (past tense verb)

______________________ (zero or more adjectives) ___________ (plural noun).

- Let `avocado` denote the language $\{$`avocado`$\}$.
- Let $noun =$ `avocado` $\cup$ `beach` $\cup$ `carrot` $\cup$ `caterpillar` $\cup$ `pencil` $\cup$ `penguins` $\cup$ `zombie`.
- Let $pluralNoun = noun$ `s`.
- Let $verb =$ `add` $\cup$ `compile` $\cup$ `eat` $\cup$ `sing` $\cup$ `swim` $\cup$ `walk`.
- Let $pastVerb = verb$ `ed`.
- Let $adjective =$ `beautiful` $\cup$ `big` $\cup$ `cold` $\cup$ `considerable` $\cup$ `furry` $\cup$ `insipid` $\cup$ `yellow`.
- Now, our Madlib™ is

  `Once upon a pencil , there was a carrot , that walked beautiful, considerable penguins.`

Regular Expressions

- A regular expression, α, is

R	$L(R)$	where
$\emptyset$	$\emptyset$	
ϵ	$\{\epsilon\}$	
$\mathbf{c}$	$\{\mathbf{c}\}$	$c \in \Sigma$
$R_1 \cup R_2$	$L(R_1) \cup L(R_2)$	R_1 and R_2 are regular expressions
$R_1 \circ R_2$	$L(R_1) \circ L(R_2)$	R_1 and R_2 are regular expressions
R_1^*	$L(R_1)^*$	R_1 is a regular expression

- Language union, concatenation, and asteration were defined in the Sept. 15 notes and Sipser p. 44.

Regular Expressions Examples

Let $\Sigma = \{\mathsf{a}, \mathsf{b}\}$.

- a^*b^* – the set of all string with zero or more a's followed by zero or more b's. For example, the strings ϵ, a, aaab, bb, and aabbb are in this language. The strings aba and ba are not.
- $(aaa)^*(bb)^*b$ – the set of all strings consisting of a number of a's that is divisible by three followed by an odd number of b's. For example, the strings b, aaabbb, and aaaaaaaaaaaabbbbb are in this language, but the strings ϵ, baaa, and aabbb are not.
- $a\Sigma^*b$ – the set of all strings that begin with an a and end with a b. For example, the strings ab, ababab and abbbaabaaabab are in this language, but the strings a, aba, and babbab are not.

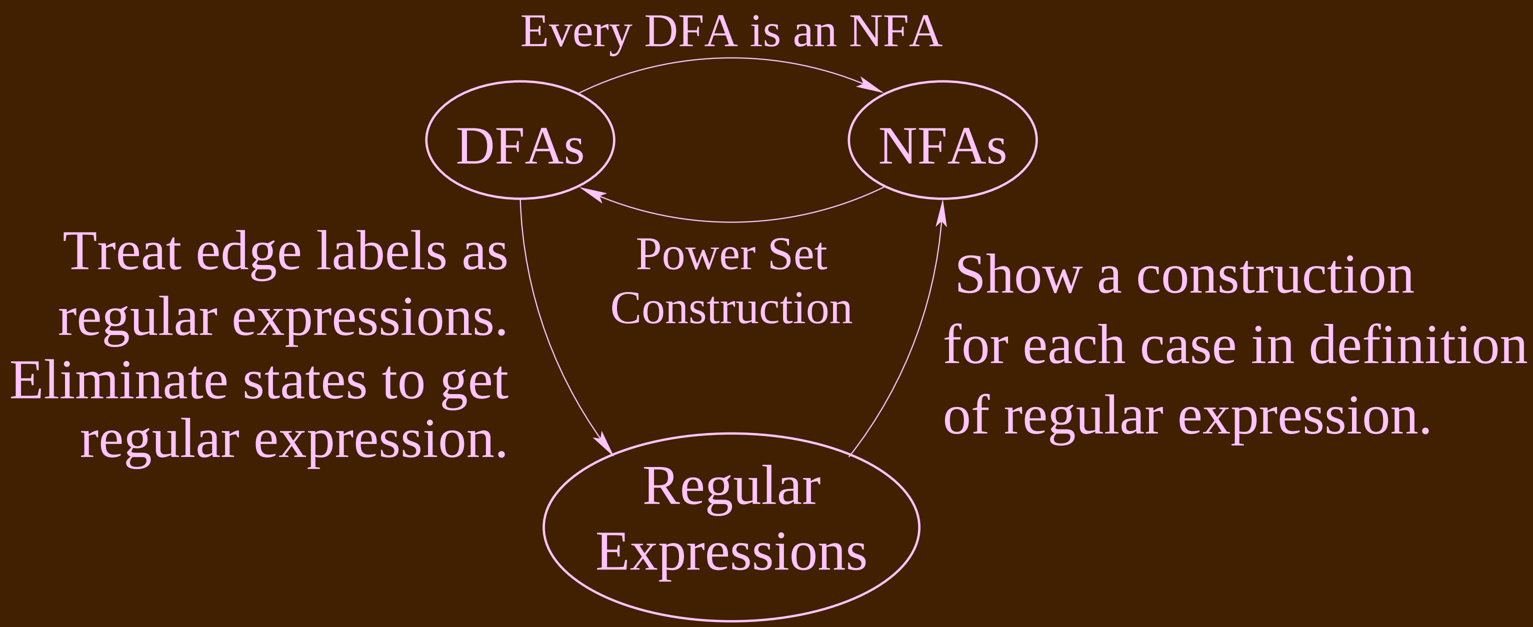
A Few More Remarks

- We'll write Σ as a regular language that generates the language of all strings in Σ^1.
- From the definition of L^*, we note that $\epsilon \in L^*$ for any language L. In particular, note that $\emptyset^* = \{\epsilon\}$.
- Regular expressions and programming languages.
  The following regular expressions describe various lexical pieces of Java:
  - The keyword `class`: `class`.
  - Identifiers: $([\mathtt{A}-\mathtt{Z}] \cup [\mathtt{a}-\mathtt{z}] \cup \_ \cup \$)([\mathtt{A}-\mathtt{Z}] \cup [\mathtt{a}-\mathtt{z}] \cup \_ \cup \$ \cup [\mathtt{0}-\mathtt{9}])$, where $[\mathtt{A}-\mathtt{Z}]$ denotes all characters from A to Z, and likewise for $[\mathtt{a}-\mathtt{z}]$ and $[\mathtt{0}-\mathtt{9}]$.
  - Floating point numbers:

$$\begin{array}{ll} & (([\mathtt{0}-\mathtt{9}]^+ \,.\, [\mathtt{0}-\mathtt{9}]^*) \cup ([\mathtt{0}-\mathtt{9}]^* \,.\, [\mathtt{0}-\mathtt{9}]^+))(\epsilon \cup (\mathtt{e}(+ \cup - \cup \epsilon)[\mathtt{0}-\mathtt{9}]^+)) \\ \bigcup & [\mathtt{0}-\mathtt{9}]^+\mathtt{e}(+ \cup - \cup \epsilon)[\mathtt{0}-\mathtt{9}]^+)), \end{array}$$

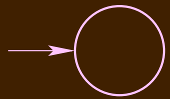
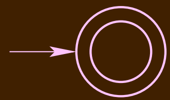
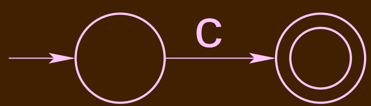
  where $[\mathtt{0}-\mathtt{9}]^+ \; = \; [\mathtt{0}-\mathtt{9}][\mathtt{0}-\mathtt{9}]^*$.

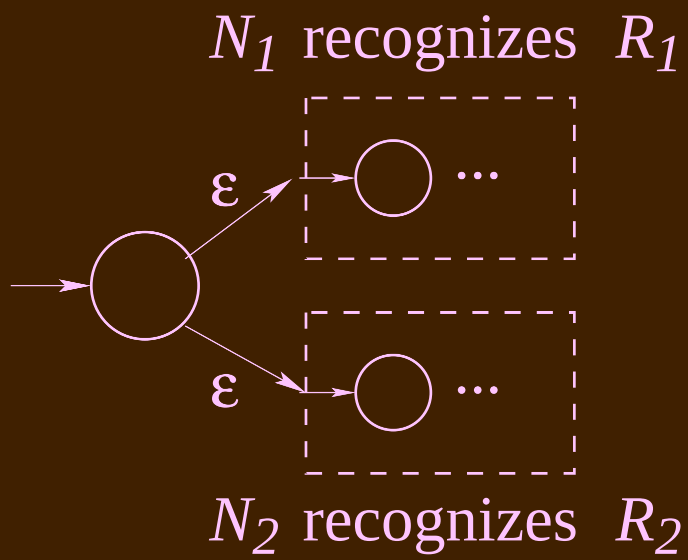
Regular Expressions = Finite Automata



- We will show that every language described by a regular expression is recognized by an NFA.
- We will then show that every language recognized by a DFA has a corresponding regular expression.

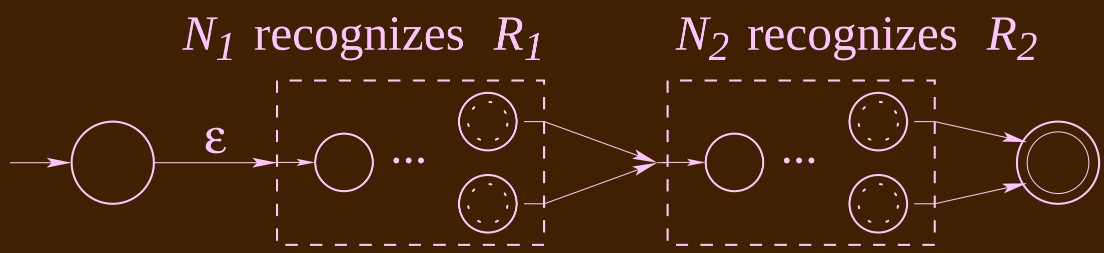
From REs to NFAs

- $R = \emptyset$: 
- $R = \epsilon$: 
- $R = \mathbf{c}$: 

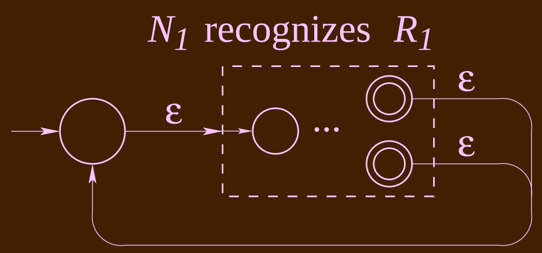
- $R = R_1 \cup R_2$: 

From REs to NFAs (cont.)

- $R = R_1 \circ R_2$:

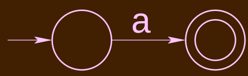


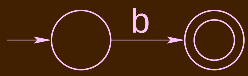
- $R = R_1^*$:

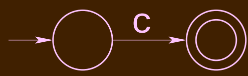


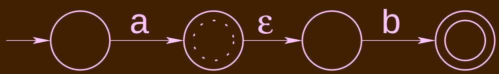
An Example

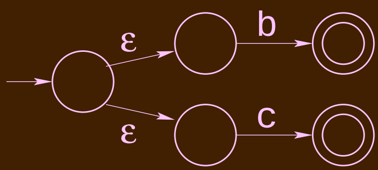
$R = (\mathsf{b} \cup \mathsf{c} \cup \mathsf{ab})^*$

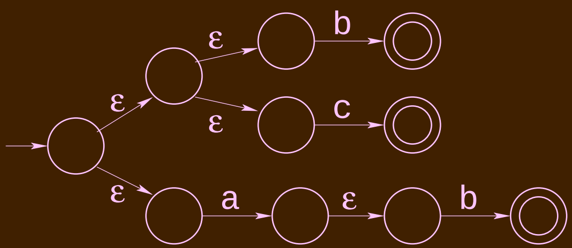
- $\mathsf{a} \equiv$ 

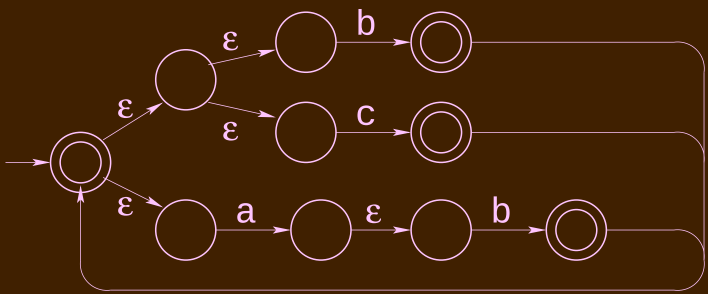
| $\mathsf{b} \equiv$ 

| $\mathsf{c} \equiv$ 

- $\mathsf{ab} \equiv$ 

- $\mathsf{b} \cup \mathsf{c} \equiv$ 

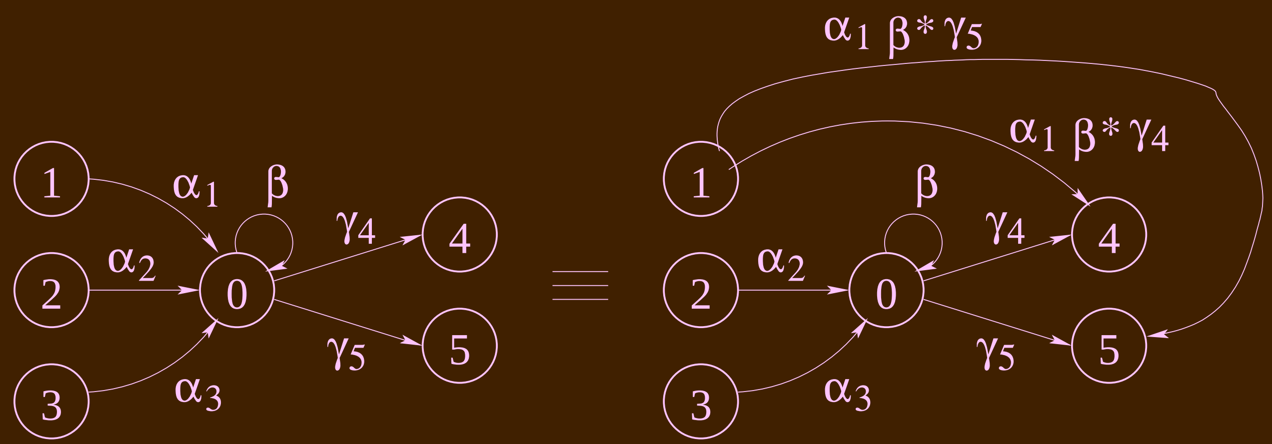
- $\mathsf{b} \cup \mathsf{c} \cup \mathsf{ab} \equiv$ 

- $(\mathsf{b} \cup \mathsf{c} \cup \mathsf{ab})^* \equiv$ 

From DFAs to REs

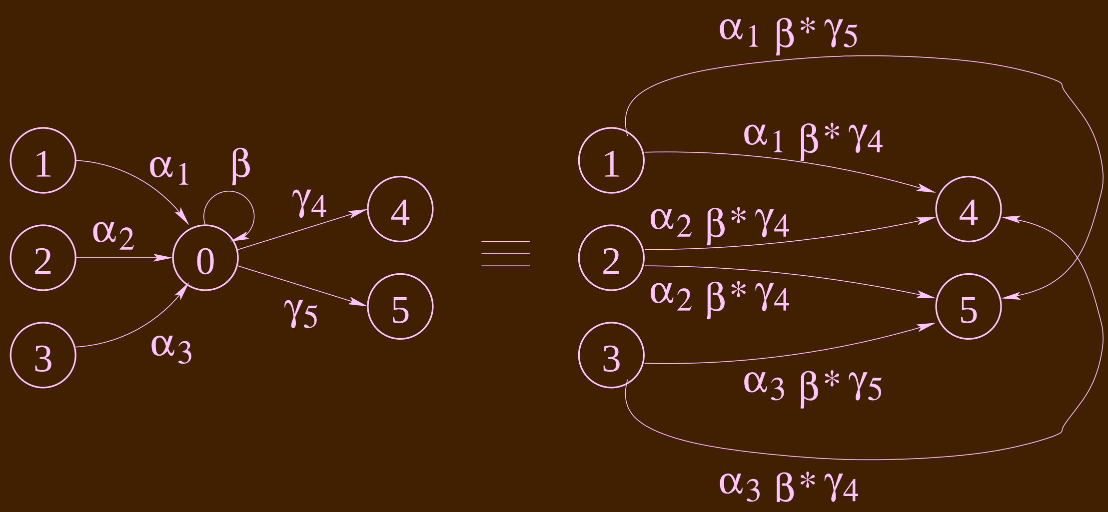
- Given a DFA, we want to construct a regular expression that for the DFA's language.
- The "hard" part is keeping track of all of the possible paths from the start state to an accepting state, especially because there can be many possible loops.
- The key observation is that the symbols that label edges in a DFA are simple regular expressions.
  - We'll generalize this idea and allow arbitrary regular expressions on edges.
  - We'll use the flexibility of regular expressions to allow us to eliminate one state from the DFA at a time. We'll modify the REs for the remaining edges to account for the deleted states. Thus, our new DFA will recognize the same language as the original one.
  - By successively deleting states, we'll eventually get to a DFA with a start state, an accept state, and a single edge from the start state to the accept state. The label for this edge is the RE corresponding to the original DFA.

Eliminating Edges (Example)



- Consider paths from state 1 to state 4 that go through state 0.
- Any such path must begin with a string that takes it to state 0 for the first time. α_1 describes such strings.
- Then, the path can visit state 0 several times. The expression β^* describes all such looping.
- Finally, the path has visited state 0 for the last time and goes to state 4. The expression γ_4 describes that part of the path.
- Thus, the set of strings that start in state 1, pass through state 0 at least once, and end in state 4 are described by the expression $\alpha_1\beta^*\gamma_4$.

Eliminating Edges (cont)



- We can replace all edges in and out of state 0 in the same way as we replaced the edge from state 1.
- Once we've done this, we can eliminate state 0 from the machine.
- The resulting machine accepts the same language as the original machcine.
- We continue, until the we have eliminated all states except for the start and accept states. The final machine accepts the same language as the original machine. The final machine has one edge whose label is the regular expression corresponding to the original DFA.

From DFAs to REs (proof 1/3)

To make a complete proof out of the preceeding observations, we define the automata that we use that have regular expressions for edge labels.

- A GNFA, G, is a 5-tuple (Q, Σ, E, s, t).
- Q is a finite set of states.
- Σ is a finite set of symbols.
- $E : Q \times Q \rightarrow$ regular expression, is the edge labeling.
- s is the start state, there are no edges going into s.
- t is the accepting state, there are no edges going out of t.
- G accepts w iff there are strings $x_1, x_2, \ldots x_k$ and states $q_1, q_1, \ldots q_{k-1}$ such that x_1 matches the regular expression for (s, q_1), x_i matches the label for (q_{i-1}, q_i), and x_k matches the label for (q_{k-1}, t).

From DFAs to REs (proof 2/3)

Given a DFA, $M = (Q_D, \Sigma, \delta_D, q_{0,D}, F_D)$, we construct a GNFA with $G = (Q_G, \Sigma, E, q_{start}, q_{accept})$ where

- $Q_G = Q_D \cup \{q_{start}, q_{accept}\}$ – we require $q_{start}, q_{accept} \notin Q_D$.
- If for each $c \in C_{i,j}$, $\delta(q_i, c) = q_j$, then E has an edge from q_i to q_j labeled with the regular expression $\bigcup_{c \in C_{i,j}} c$.
- There is an edge from q_{start} to $q_{0,D}$ labeled with ϵ.
- There is an edge from each state in F_D to q_{accept}, and each such edge is labeled with ϵ.
- By this construction, $L(G) = L(M)$.

From DFAs to REs (proof 3/3)