CS311—Computational Structures

Regular Languages and Regular Expressions

Lecture 4

Andrew P. Black
Andrew Tolmach

Expressions

- We're used to using **expressions** to describe mathematical objects
  - Example: the arithmetic expression (2*11)+20 describes the value 42
- Expressions are useful because they are precise, compact, and (often) can be simplified using algebraic laws
- **Regular expressions** describe languages (sets of strings) over an alphabet

Regular Expressions

- The regular expressions over an alphabet Σ are defined **inductively**:
  - Base cases:
    - *a* is an r.e., for each $a \in \Sigma$
    - ε is an r.e.
    - ∅ is an r.e.
  - Inductive cases:
    - $(R_1 . R_2)$ is an r.e. when R_1,R_2 are r.e.'s
    - $(R_1 + R_2)$ is an r.e. when R_1,R_2 are r.e.'s
    - (R^*) is an r.e. when R is a r.e.
  - Nothing else is an r.e.

The meaning of an r.e.

- Each r.e. corresponds to a language.
- We'll write $\mathscr{L}$ for the function that maps each r.e. to its corresponding language

$\mathscr{L}[\ a\ ] = \{a\}$ $\mathscr{L}[\ \varepsilon\ ] = \{\varepsilon\}$

$\mathscr{L}[\varnothing] = \{\ \}$

$\mathscr{L}[(p\ .\ q)] = \mathscr{L}[p]\ .\ \mathscr{L}[q]$ (concatenation)

$\mathscr{L}[(p + q)] = \mathscr{L}[p] \cup \mathscr{L}[q]$ (union)

$\mathscr{L}[(p^*)\ ] = \mathscr{L}[p]^*$ (0 or more from $\mathscr{L}[p]$)

Portland State UNIVERSITY

Examples

1. $\mathcal{L}[\ ((a\ .\ (b^*))\ +\ c)\ ]$ =

2. $\mathcal{L}[\ (((a + b)\ .\ (a + b))^*)\ ]$ =

3. $\mathcal{L}[\ (((\varepsilon + b)\ .\ ((a\ .\ b)^*))\ .\ (\varepsilon + a))\ ]$ =

4. $\mathcal{L}[\ (a\ .\ \varnothing)\ ]$ =

Examples

1. $\mathscr{L}[\![((a \,.\, (b^*)) + c)]\!] =$

   $L_I = \{a, ab, abb, abbb, \ldots, c\}$

2. $\mathscr{L}[\![(((a + b) \,.\, (a + b))^*)]\!] =$

3. $\mathscr{L}[\![(((\varepsilon + b) \,.\, ((a \,.\, b)^*)) \,.\, (\varepsilon + a))]\!] =$

4. $\mathscr{L}[\![(a \,.\, \varnothing)]\!] =$

Examples

1. $\mathscr{L}[\![((a \,.\, (b^*)) + c)]\!] =$

   $L_1 = \{a, ab, abb, abbb, \ldots, c\}$

2. $\mathscr{L}[\![(((a + b) \,.\, (a + b))^*)]\!] =$

   $L_2 = \{\varepsilon, aa, ab, ba, abab, bbaa, baabbaabab, \ldots\}$

3. $\mathscr{L}[\![(((\varepsilon + b) \,.\, ((a \,.\, b)^*)) \,.\, (\varepsilon + a))]\!] =$

4. $\mathscr{L}[\![(a \,.\, \varnothing)]\!] =$

Examples

1. $\mathscr{L}[\![((a \, . \, (b^*)) + c)]\!] =$

   $L_1 = \{a, ab, abb, abbb, \ldots, c\}$

2. $\mathscr{L}[\![(((a + b) \, . \, (a + b))^*)]\!] =$

   $L_2 = \{\varepsilon, aa, ab, ba, abab, bbaa, baabbaabab, \ldots\}$

3. $\mathscr{L}[\![(((\varepsilon + b) \, . \, ((a \, . \, b)^*)) \, . \, (\varepsilon + a))]\!] =$

   $\{\varepsilon, \quad \}$

4. $\mathscr{L}[\![(a \, . \, \varnothing)]\!] =$

Examples

1. $\mathscr{L}[\![((a \,.\, (b^*)) + c)]\!] =$

   $L_1 = \{a, ab, abb, abbb, \ldots, c\}$

2. $\mathscr{L}[\![(((a + b) \,.\, (a + b))^*)]\!] =$

   $L_2 = \{\varepsilon, aa, ab, ba, abab, bbaa, baabbaabab, \ldots\}$

3. $\mathscr{L}[\![(((\varepsilon + b) \,.\, ((a \,.\, b)^*)) \,.\, (\varepsilon + a))]\!] =$

   $\{\varepsilon, ab, \qquad \}$

4. $\mathscr{L}[\![(a \,.\, \varnothing)]\!] =$

Examples

1. $\mathscr{L}[\![((a\ .\ (b^*)) + c)]\!] =$

   $L_1 = \{a, ab, abb, abbb, \ldots, c\}$

2. $\mathscr{L}[\![(((a + b)\ .\ (a + b))^*)]\!] =$

   $L_2 = \{\varepsilon, aa, ab, ba, abab, bbaa, baabbaabab, \ldots\}$

3. $\mathscr{L}[\![(((\varepsilon + b)\ .\ ((a\ .\ b)^*))\ .\ (\varepsilon + a))]\!] =$

   $\{\varepsilon, ab, abab, \qquad\qquad \}$

4. $\mathscr{L}[\![(a\ .\ \varnothing)]\!] =$

Portland State UNIVERSITY

Examples

1. $\mathscr{L}[\![((a \,.\, (b^*)) + c)]\!] =$

   $L_1 = \{a, ab, abb, abbb, \ldots, c\}$

2. $\mathscr{L}[\![(((a + b) \,.\, (a + b))^*)]\!] =$

   $L_2 = \{\varepsilon, aa, ab, ba, abab, bbaa, baabbaabab, \ldots\}$

3. $\mathscr{L}[\![(((\varepsilon + b) \,.\, ((a \,.\, b)^*)) \,.\, (\varepsilon + a))]\!] =$

   $\{\varepsilon, ab, abab, ababab, \qquad \}$

4. $\mathscr{L}[\![(a \,.\, \varnothing)]\!] =$

Examples

1. $\mathscr{L}[\![((a \,.\, (b^*)) + c)]\!] =$

   $L_1 = \{a, ab, abb, abbb, \ldots, c\}$

2. $\mathscr{L}[\![(((a + b) \,.\, (a + b))^*)]\!] =$

   $L_2 = \{\varepsilon, aa, ab, ba, abab, bbaa, baabbaabab, \ldots\}$

3. $\mathscr{L}[\![(((\varepsilon + b) \,.\, ((a \,.\, b)^*)) \,.\, (\varepsilon + a))]\!] =$

   $\{\varepsilon, ab, abab, ababab, bab, \quad\quad \}$

4. $\mathscr{L}[\![(a \,.\, \varnothing)]\!] =$

Portland State UNIVERSITY

Examples

1. $\mathscr{L}[\![((a \,.\, (b^*)) + c)]\!] =$

   $L_1 = \{a, ab, abb, abbb, \ldots, c\}$

2. $\mathscr{L}[\![(((a + b) \,.\, (a + b))^*)]\!] =$

   $L_2 = \{\varepsilon, aa, ab, ba, abab, bbaa, baabbaabab, \ldots\}$

3. $\mathscr{L}[\![(((\varepsilon + b) \,.\, ((a \,.\, b)^*)) \,.\, (\varepsilon + a))]\!] =$

   $\{\varepsilon, ab, abab, ababab, bab, baba, \qquad \}$

4. $\mathscr{L}[\![(a \,.\, \varnothing)]\!] =$

Portland State UNIVERSITY

Examples

1. $\mathscr{L}[\![((a \,.\, (b^*)) + c)]\!] =$

   $L_1 = \{a, ab, abb, abbb, \ldots, c\}$

2. $\mathscr{L}[\![(((a + b) \,.\, (a + b))^*)]\!] =$

   $L_2 = \{\varepsilon, aa, ab, ba, abab, bbaa, baabbaabab, \ldots\}$

3. $\mathscr{L}[\![(((\varepsilon + b) \,.\, ((a \,.\, b)^*)) \,.\, (\varepsilon + a))]\!] =$

   $\{\varepsilon, ab, abab, ababab, bab, baba, aba, \ldots\}$

4. $\mathscr{L}[\![(a \,.\, \varnothing)]\!] =$

Examples

1. $\mathscr{L}[\![((a \,.\, (b^*)) + c)]\!] =$

   $L_1 = \{a, ab, abb, abbb, \ldots, c\}$

2. $\mathscr{L}[\![(((a + b) \,.\, (a + b))^*)]\!] =$

   $L_2 = \{\varepsilon, aa, ab, ba, abab, bbaa, baabbaabab, \ldots\}$

3. $\mathscr{L}[\![(((\varepsilon + b) \,.\, ((a \,.\, b)^*)) \,.\, (\varepsilon + a))]\!] =$

   $\{\varepsilon, ab, abab, ababab, bab, baba, aba, \ldots\}$

4. $\mathscr{L}[\![(a \,.\, \varnothing)]\!] =$

   $L_4 = \{\ \}$

Common Shorthands

- Concatenation (.) is usually not written
  - More precisely: written as juxtaposition
- We assign precedence to the operators and then omit parentheses if possible
  - * groups most tightly, then ., then +
  - e.g., a+bc* means (a + (b .(c*)))
- Write R^+ for RR* (one or more from R)
- Write R^k for RR...R k times (k from R)
- If our alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$, then we write Σ for the r.e. $(a_1 + a_2 + \ldots + a_n)$

Portland State UNIVERSITY

More compact examples

- $\mathcal{L}[(\varepsilon + 1)(01)^*(\varepsilon + 0)] =$
- $\mathcal{L}[\Sigma^*001\Sigma^*] =$ (assuming $\Sigma=\{0,1\}$)
- $\mathcal{L}[\quad] = \{w \in \{0,1\}^* \mid w$ starts and ends with the same symbol$\}$
- $\mathcal{L}[\quad] = \{w \in \{0,1\}^* \mid$ w contains an odd number of 0s $\}$

Portland State University

Simplifying r.e.s

- Just as for arithmetic expressions, r.e.s can be simplified by algebraic laws.
- Some useful laws:
  - $R + P = P + R$
  - $R + \varnothing = R$
  - $R\varepsilon = R = \varepsilon R$
  - $\varnothing R = \varnothing = R\varnothing$
  - $\varnothing^* = \varepsilon$

Simplifying r.e.s

- Just as for arithmetic expressions, r.e.s can be simplified by algebraic laws.
- Some useful laws:
  - $R + P = P + R$
  - $R + \varnothing = R$
  - $R\varepsilon = R = \varepsilon R$
  - $\varnothing R = \varnothing = R\varnothing$
  - $\varnothing^* = \varepsilon$

See Hein §11.1.2 and slides 27–30 for more!

Practical uses for r.e.s

- Widely used for specifying text patterns
  - typically with extended r.e. syntax for ASCII
  - e.g., unix grep command

```
%grep “^e[a-z]i[a-z]*a$” /etc/dict/words
```

  - enigma epiblastema epiblema ... eria

  - e.g., lexical analyzer generation for compilers

```
[A-Za-z_][A-Za-z_0-9]*
```

describes format of identifiers in C programs

Regular Expressions and Regular Languages are equivalent!

- That is, they describe exactly the same class of languages (hence their names)
- Must prove this in two directions:
  - Every r.e. defines a regular language
    - We've already done most of the work, so this shouldn't be too surprising
  - Every regular language is defined by an r.e.
    - This is harder

R is an r.e. => $\mathcal{L}$(R) is regular

- Claim: For each r.e. R, we can construct an NFA N that recognizes $\mathcal{L}$(R)
  - We can then convert N to a DFA M recognizing $\mathcal{L}$(R), so $\mathcal{L}$(R) is regular
- Proof is by **structural induction** on R
  - One case for each rule for constructing R
  - Inductive hypothesis is: if claim is true for each sub-expression of R, then it's true for R itself

Proof Outline: Six Cases

- $R = a$ for some a in Σ. Then $\mathcal{L}(R) = \{a\}$. So...
- $R = \varepsilon$. Then $\mathcal{L}(R) = \{\varepsilon\}$. So...
- $R = \varnothing$. Then $\mathcal{L}(R) = \{\}$. So...
- $R = R_1 + R_2$. Then $\mathcal{L}(R) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$.
  - By the inductive hypothesis we can construct NFA's N_1 recognizing $\mathcal{L}(R_1)$ and N_2 recognizing $\mathcal{L}(R_2)$. So...
- $R = R_1 \,.\, R_2$. Then $\mathcal{L}(R) = \mathcal{L}(R_1) \,.\, \mathcal{L}(R_2)$.
  - By the inductive hypothesis...
- $R = (R_1)^*$. Then $\mathcal{L}(R) = (\mathcal{L}(R_1))^*$. By...

L is recognized by a DFA ⇒ ∃ an r.e. R such that $\mathcal{L}(R) = L$

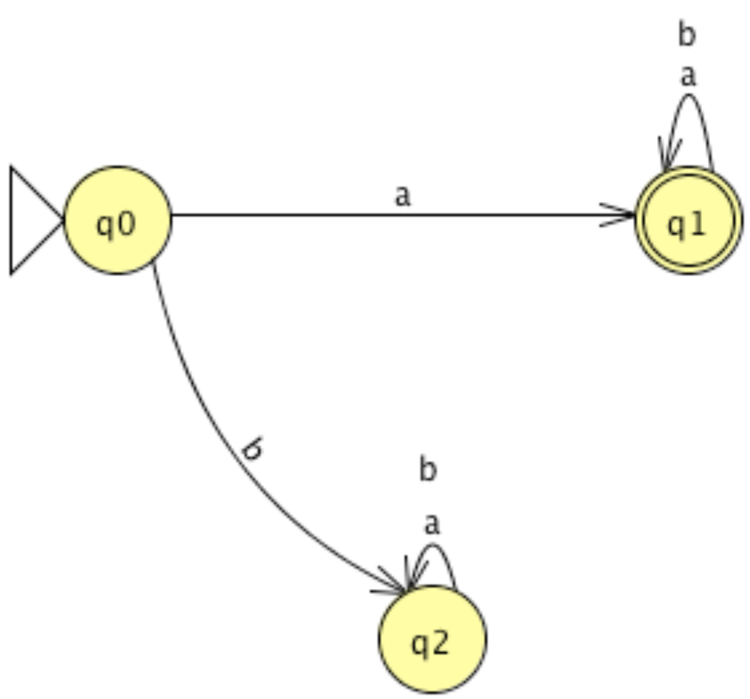
- Challenge: start with an arbitrary DFA and find a corresponding r.e.
  - There's more than one way to do this (see IALC)
- 1st idea: use generalization of NFAs in which transitions can be labeled by r.e.s.

NFA ⇒ r.e. by State Elimination

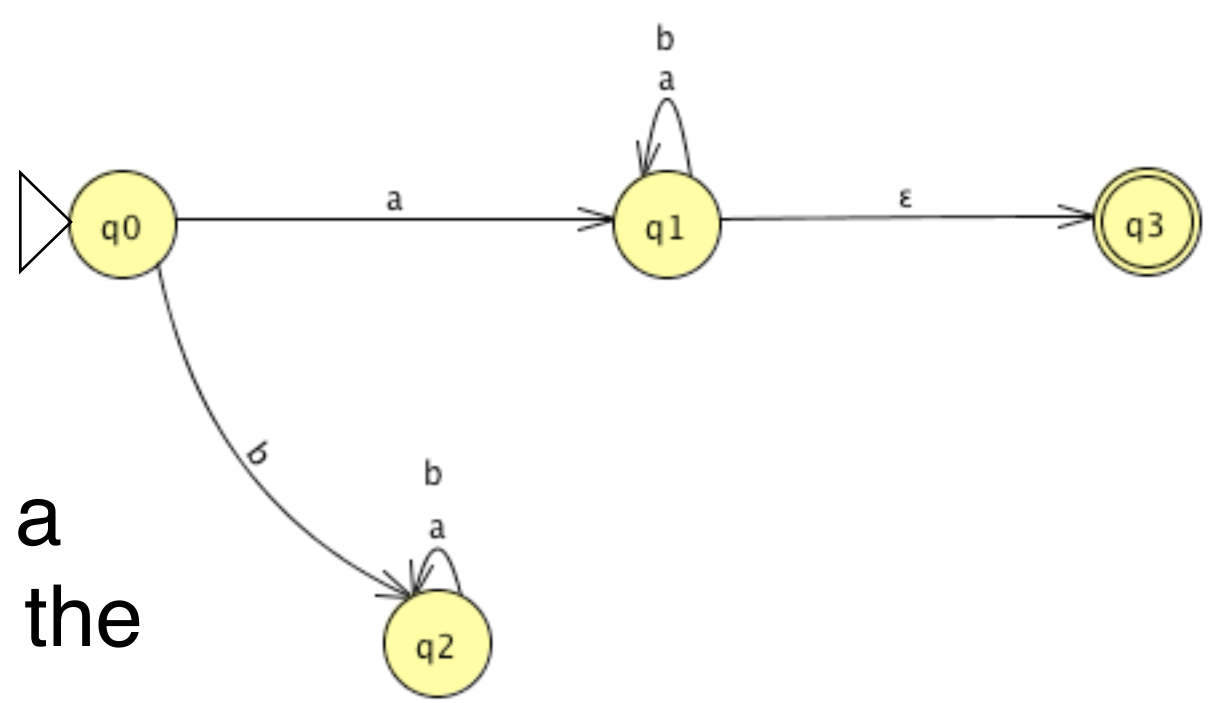
- Allow the labels on an NFA's transitions to be r.e.s rather than just single symbols.
  - Any string that is in the language of the r.e. enables the transition.
- Remove states one at a time, keeping language the same by making labels more complex
- Ultimately, machine has one transition; label is desired r.e. for original machine

Example

0. If there is no arc from state *i* to state *j*, imagine one with label ∅.

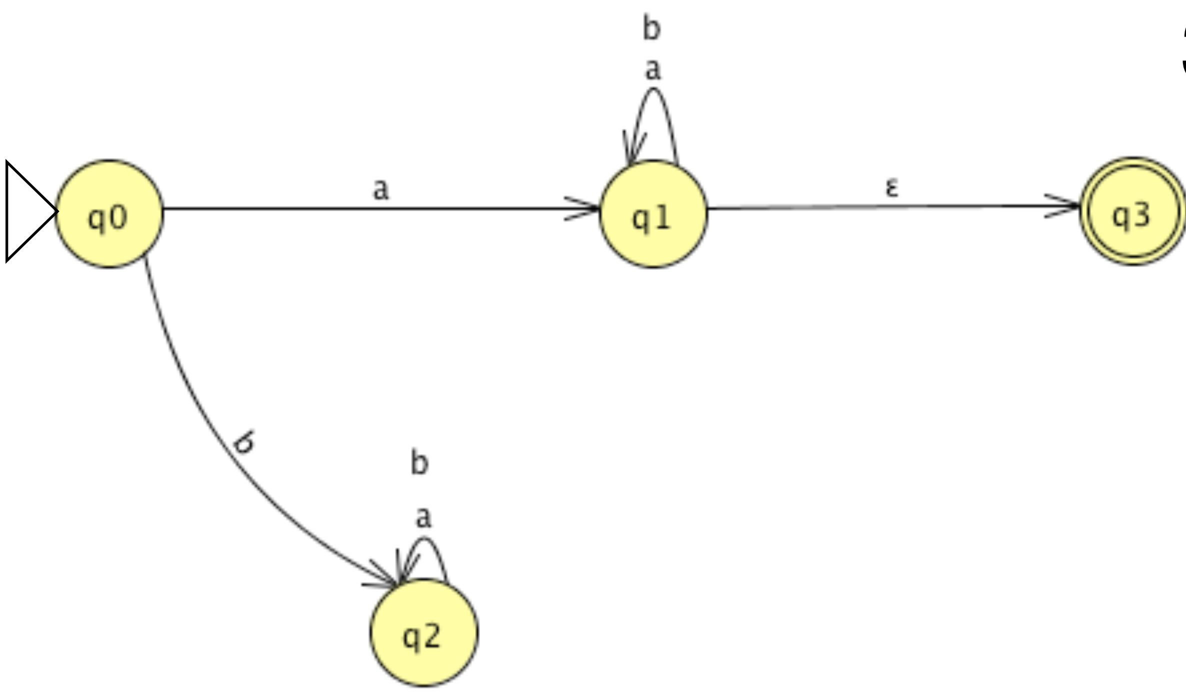


1. If the initial state has a self-transition, create a new initial state with a single ε-transition to the old initial state.

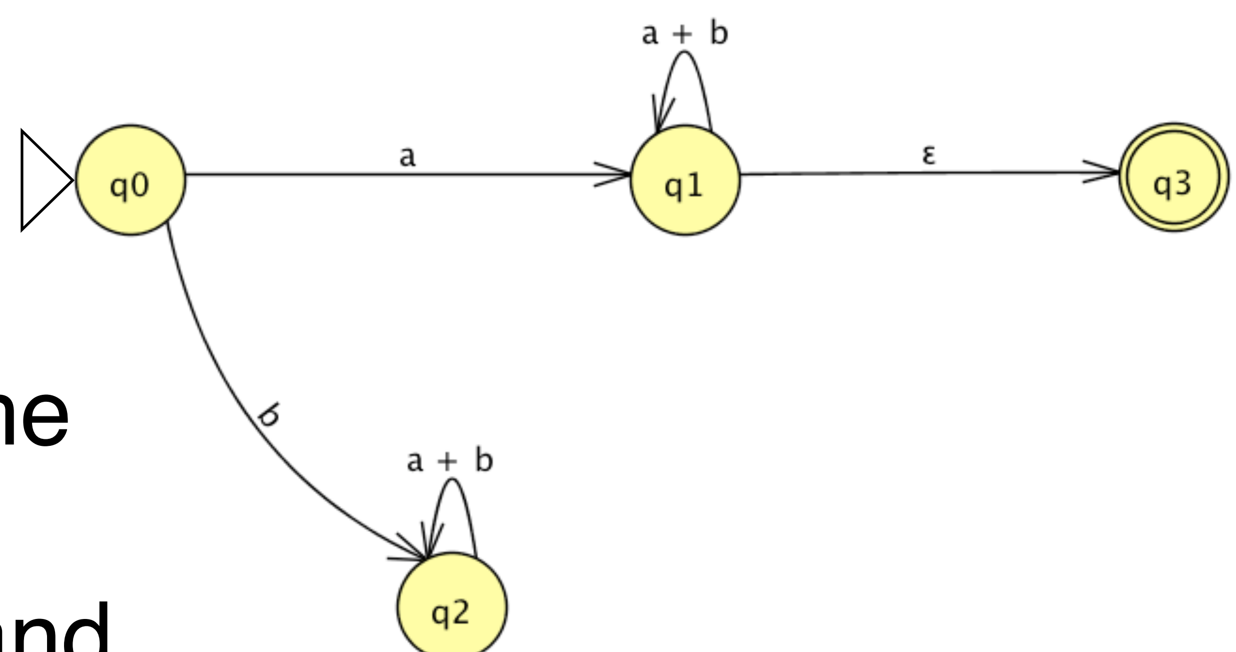


2. Create a new final state with a ε-transition to it from each of the old final states.

Example



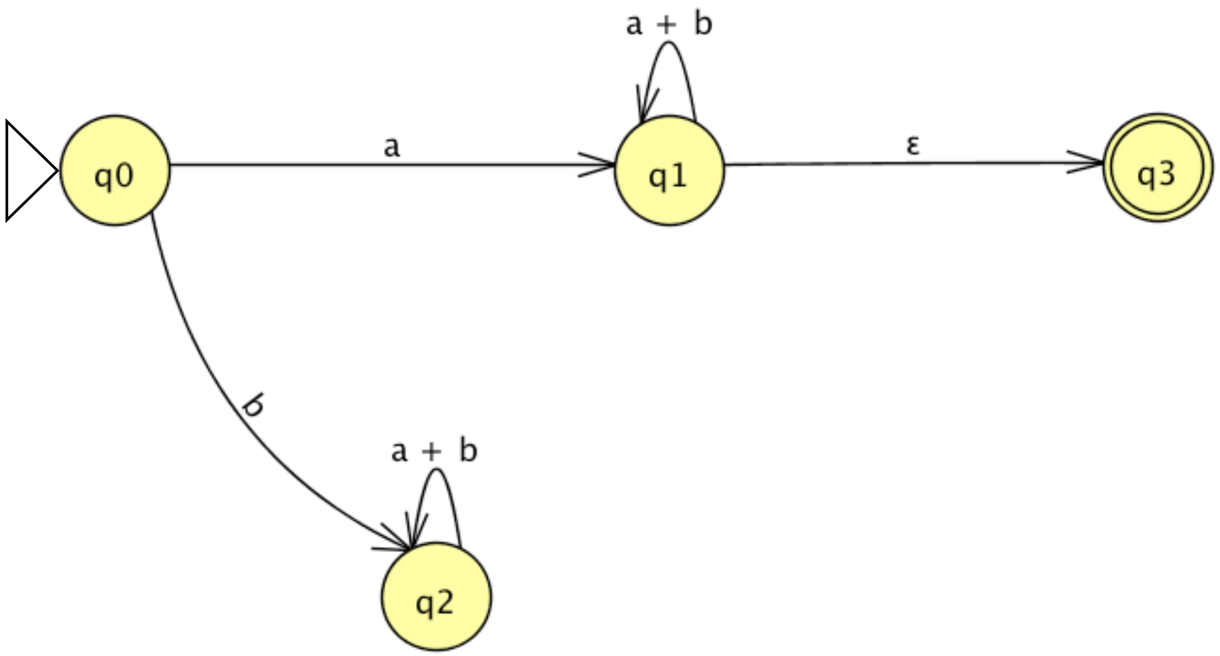
3. For each pair of states *i*, *j* with more than one transition from *i* to *j*, replace them all by a single transition labeled with the r.e. that is the sum of the old labels.



4. Eliminate one state at a time until the only states that remain are the start state and the final state:

Portland State UNIVERSITY

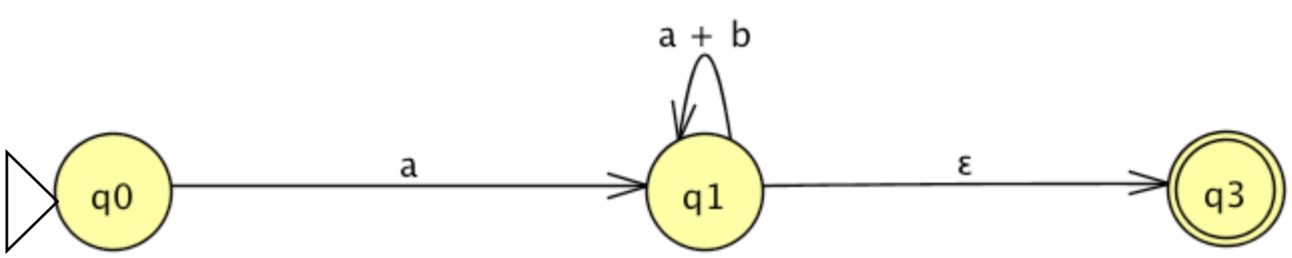
How to Eliminate State *k*



- For each pair of nodes *i, j* (*i* ≠ *k*, *j* ≠ *k*), label the transition from *i* to *j* with:

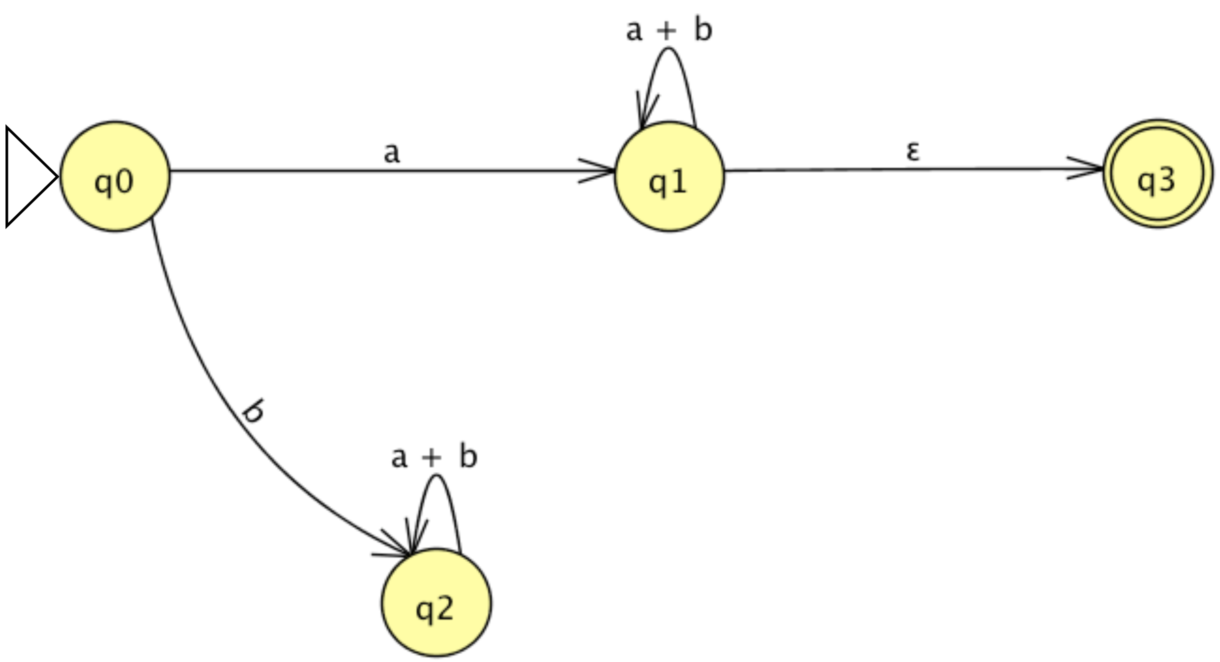
$$(i, j) + (i, k)(k, k)^*(k, j)$$

- Remove state *k* and all its transitions.



Portland State University

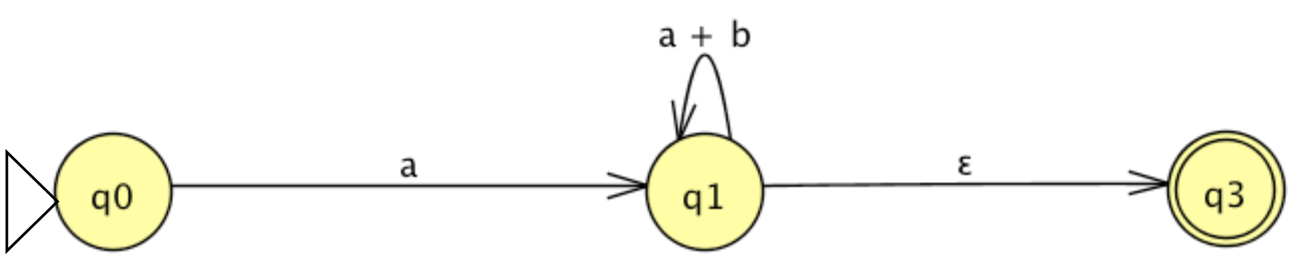
How to Eliminate State *k*

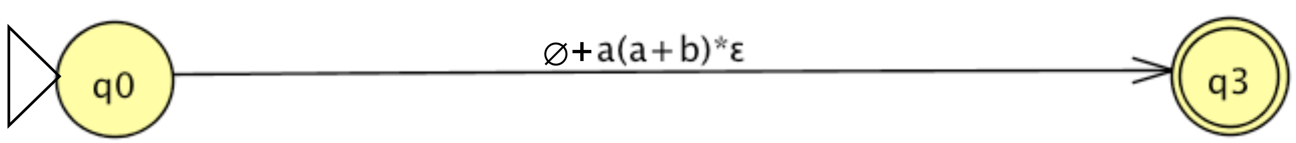


- For each pair of nodes *i, j* (*i* ≠ *k*, *j* ≠ *k)*, label the transition from *i* to *j* with:

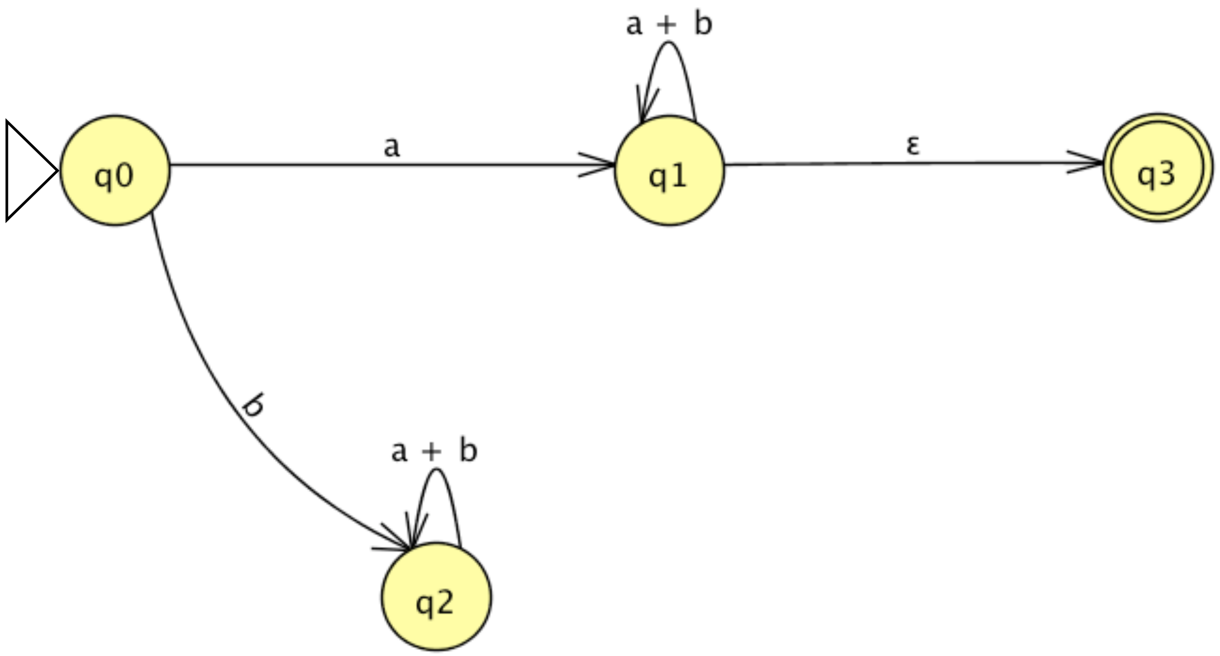
$$(i, j) + (i, k)(k, k)^*(k, j)$$

- Remove state *k* and all its transitions.





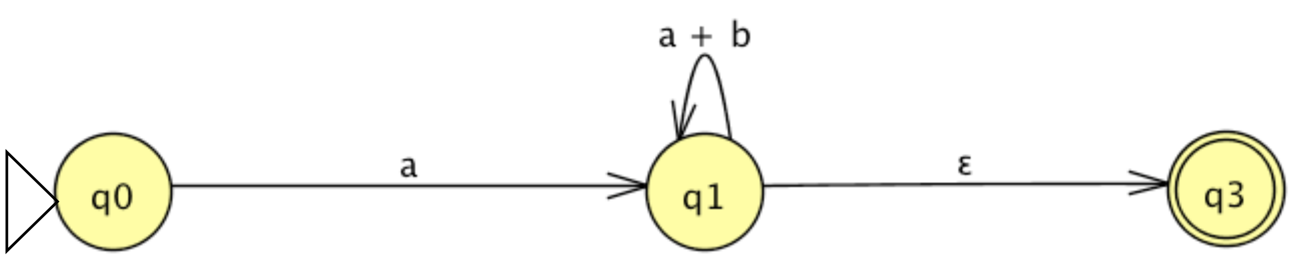
How to Eliminate State *k*

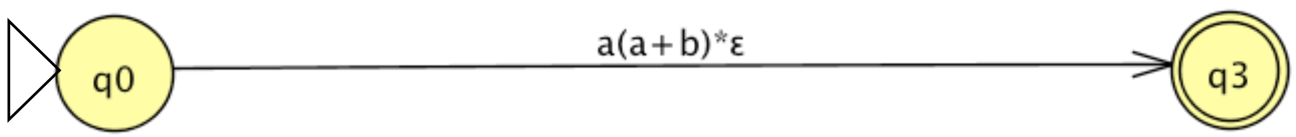


- For each pair of nodes *i, j* (*i* ≠ *k*, *j* ≠ *k)*, label the transition from *i* to *j* with:

$$(i, j) + (i, k)(k, k)^*(k, j)$$

- Remove state *k* and all its transitions.





Portland State University

The Algorithm from Hein:

Finite Automaton to Regular Expression (11.5)

Assume that we have a DFA or an NFA. Perform the following steps:

1. Create a new start state s, and draw a new edge labeled with Λ from s to the original start state.
2. Create a new final state f, and draw new edges labeled with Λ from all the original final states to f.
3. For each pair of states i and j that have more than one edge from i to j, replace all the edges from i to j by a single edge labeled with the regular expression formed by the sum of the labels on each of the edges from i to j.
4. Construct a sequence of new machines by eliminating one state at a time until the only states remaining are s and f. As each state is eliminated, a new machine is constructed from the previous machine as follows:

The Algorithm from Hein:

Finite Automaton to Regular Expression (11.5)

Assume that we have a DFA or an NFA. Perform the following steps:

1. Create a new start state s, and draw a new edge labeled with ε from s to the original start state.
2. Create a new final state f, and draw new edges labeled with ε from all the original final states to f.
3. For each pair of states i and j that have more than one edge from i to j, replace all the edges from i to j by a single edge labeled with the regular expression formed by the sum of the labels on each of the edges from i to j.
4. Construct a sequence of new machines by eliminating one state at a time until the only states remaining are s and f. As each state is eliminated, a new machine is constructed from the previous machine as follows:

Eliminate State k

For convenience we'll let old(i, j) denote the label on edge (i, j) of the current machine. If there is no edge (i, j), then set old(i, j) = ∅. Now for each pair of edges (i, k) and (k, j), where $i \neq k$ and $j \neq k$, calculate a new edge label, new(i, j), as follows:

$$\text{new}(i, j) = \text{old}(i, j) + \text{old}(i, k)\ \text{old}(k, k)^*\ \text{old}(k, j).$$

For all other edges (i, j) where $i \neq k$ and $j \neq k$, set

$$\text{new}(i, j) = \text{old}(i, j).$$

The states of the new machine are those of the current machine with state k eliminated. The edges of the new machine are the edges (i, j) for which label new(i, j) has been calculated.

Now s and f are the two remaining states. If there is an edge (s, f), then the regular expression new(s, f) represents the language of the original automaton. If there is no edge (s, f), then the language of the original automaton is empty, which is signified by the regular expression ∅.

Essentially the same algorithm is in Hopcroft et. al. § 3.2.2

From DFA to r.e. by paths

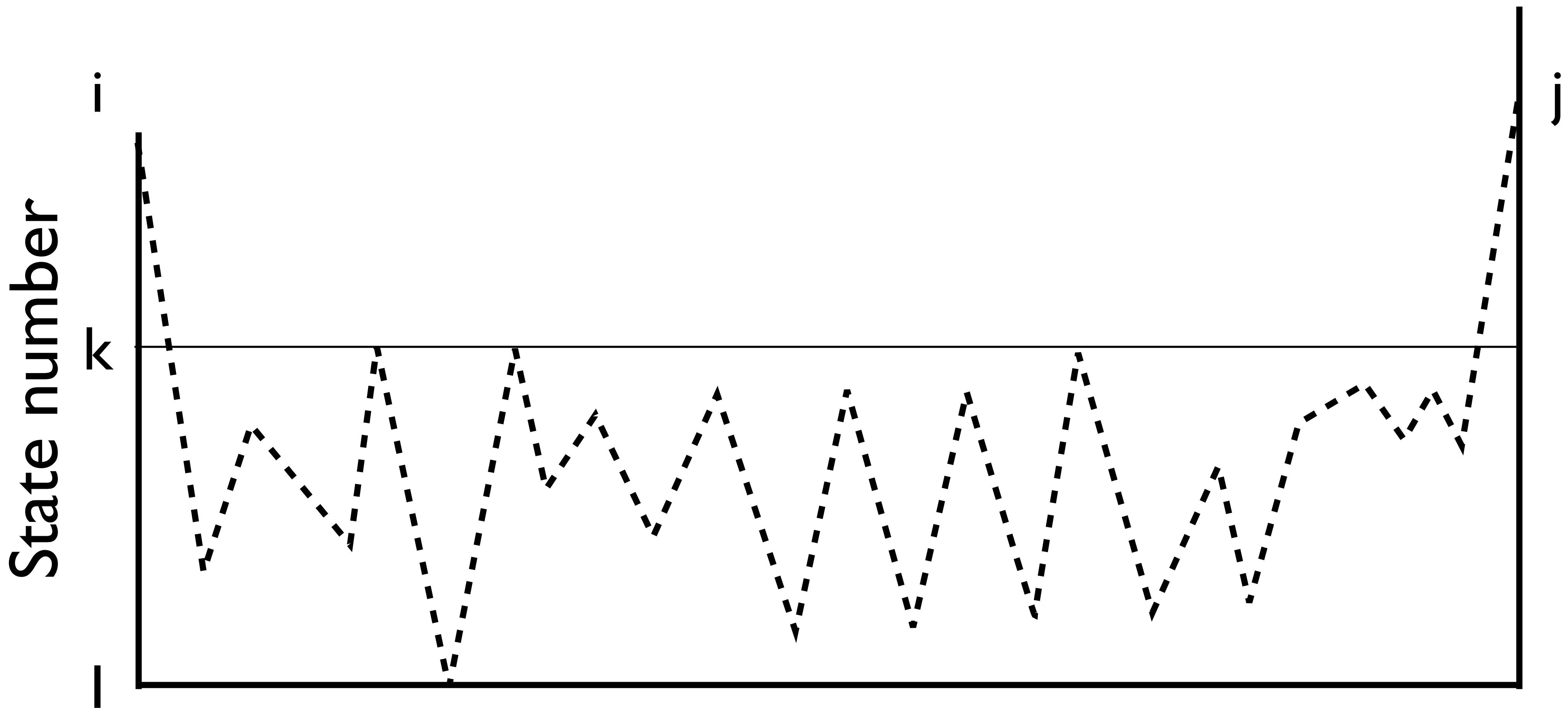
- 2nd idea: r.e.s correspond to ***paths*** in DFA
    - The language recognized by a DFA M is the set of strings accepted by M.
    - Each accepted string defines a *path* through M from the start state to some final state.
    - Show how to construct an r.e. corresponding to **any** path in the DFA, using induction.
    - Combine appropriate path r.e.s to build an r.e. for the *accepting* paths

Inductive path definitions

- Assume M's states are named 1,2,...,n.
- Define R_{ij} = an r.e. whose language is {w | w drives M from state i to state j}

If start state = s and final states = $\{f_1, f_2, \ldots, f_m\}$, then r.e. for M is $R = R_{sf_1} + R_{sf_2} + \ldots + R_{sf_m}$

- To set-up the induction: let $R_{ij}^{(k)}$ = an r.e. whose language is {w | w drives M from state i to state j without going through any intermediate state > k}
  - Note that path endponts i,j **are** allowed to be > k
  - We'll construct $R_{ij}^{(k)}$ by induction on k.
  - $R_{ij} = R_{ij}^{(n)}$ represents **all** paths from i to j



- A path from state i to state j that does not pass through any state > k

Base case: define $R^{(0)}$

- Since all states are numbered 1 or above, the paths in this case must have no intermediate states at all.
  - If $i \neq j$, path must have length 1 and be a single transition from state i to state j
    - Here $R_{ij}^{(0)} = \emptyset + a_1 + a_2 + \ldots + a_n$, where $a_1, \ldots, a_n$ are the labels of all transitions from state i to state j
  - If $i = j$, path may have length 0 or 1
    - Here $R_{ii}^{(0)} = \varepsilon + a_1 + a_2 + \ldots + a_n$, where $a_1,\ldots,a_n$ are the labels of all transitions from state i to itself

Inductive step: define $R^{(k)}$ using $R^{(k-1)}$

There are two possible cases for a path:

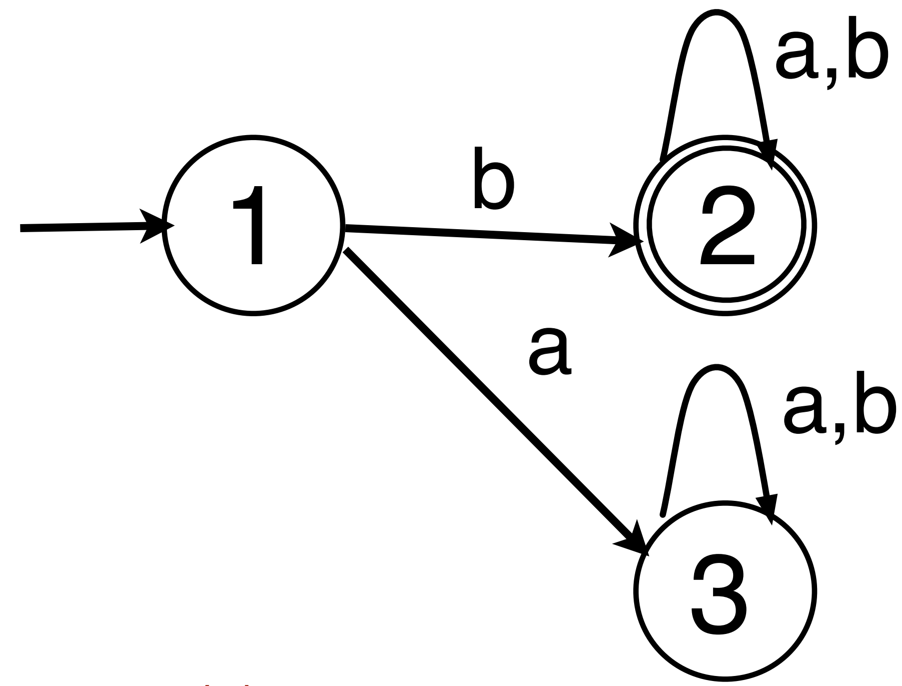
1. The path does not go through state k at all
   - Then the path is already in $R_{ij}^{(k-1)}$
2. The path goes through state k at least once
   - Then we can break it into three pieces:
     - a piece from state i to state k, described by $R_{ik}^{(k-1)}$
     - zero or more pieces going from state k back to state k (using only states lower than k), described by $(R_{kk}^{(k-1)})^*$
     - a piece from state k to state j, described by $R_{kj}^{(k-1)}$
   - The overall path is given by $R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$

- So the full r.e. is $R_{ij}^{(k-1)} + R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$

For the details …

- See Hopcroft et al. Theorem 3.2.1

Example: DFA to r.e.



$R = R_{12}^{(3)} = R_{12}^{(2)} + R_{13}^{(2)}(R_{33}^{(2)})^*R_{32}^{(2)}$!! $R_{32}^{(2)} = \varnothing$

$\therefore R = R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)}(R_{22}^{(1)})^*R_{22}^{(1)}$

$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)}(R_{11}^{(0)})^*R_{12}^{(0)} = b + \varepsilon\varepsilon^*b = b$

$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^*R_{12}^{(0)}$!! $R_{21}^{(0)} = \varnothing$

$\therefore R_{22}^{(1)} = R_{22}^{(0)} = (\varepsilon + a + b) = (a + b)$

$\therefore R = b + b(a + b)^*(a + b) = b(a+b)^*$ (why?)

Basic algebraic laws (1)

$$R \overset{\text{def}}{=} S \Leftrightarrow \mathcal{L}(R) = \mathcal{L}(S)$$

1. Union properties
   1.1. $R + T = T + R$
   1.2. $R + \varnothing = R$
   1.3. $R + R = R$
   1.4. $(R + S) + T = R + (S + T)$
2. Concatenation properties
   2.1. $R\varnothing = \varnothing R = \varnothing$
   2.2. $R\varepsilon = \varepsilon R = R$
   2.3. $(RS)T = R(ST)$
3. Distributive properties
   3.1. $R(S + T) = RS + RT$
   3.2. $(S + T)R = SR + TR$

Portland State University

Basic algebraic laws (2)

4. Kleene-* properties
    4.1. $R^* = \varepsilon + RR^* = \varepsilon + R^*R$
    4.2. If $R + ST \leq T$ then $S^*R \leq T$
    4.3. If $R + TS \leq T$ then $RS^* \leq T$

(We don't normally use these laws directly)

$$R \leq S \overset{\text{def}}{=} \mathcal{L}(R) \subseteq \mathcal{L}(S)$$

$$R \leq S \Leftrightarrow R + S = S$$

$$R = S \Leftrightarrow R \leq S \text{ and } S \leq R$$

Useful Derived Properties

5. Properties derivable from previous laws
   - 5.1. $\varnothing^* = \varepsilon^* = \varepsilon$
   - 5.2. $R^* = R^*R^* = (R^*)^* = R + R^*$
   - 5.3. $R^* = \varepsilon + R^* = (\varepsilon + R)^* = (\varepsilon + R)R^*$
   - 5.4. $R^* = (R + \ldots + R^k)^*$ for any $k \geq 1$
   - 5.5. $R^* = \varepsilon + R + \ldots + R^{k-1} + R^kR^*$ for any $k \geq 1$
   - 5.6. $R^*R = RR^*$
   - 5.7. $(R + S)^* = (R^* + S^*)^* = (R^*S^*)^* = (R^*S)^*R^* = R^*(SR^*)^*$
   - 5.8. $R(SR)^* = (RS)^*R$
   - 5.9. $(R^*S)^* = \varepsilon + (R + S)^*S$
   - 5.10. $(RS^*)^* = \varepsilon + R\,(R + S)^*$

Use laws to prove equalities

Example: prove that a*(b + ab*) = b + aa*b*.

Proof:

a*(b+ab*) = (by 3.1)

a*b + a*ab* = (by 4.1)

(ε+aa*)b + a*ab* = (by 3.1)

εb + aa*b + a*ab* = (by 2.2)

b + aa*b + a*ab* = (by 5.6)

b + aa*b + aa*b* = (by 3.1)

b + aa*(b+b*) = (by 5.2)

b + aa*b*

Use laws to prove equalities

Example: prove that a*(b + ab*) = b + aa*b*.

Proof:

3. Distributive properties
   3.1. R(S + T) = RS + RT
   3.2. (S + T)R = SR + TR

a*(b+ab*) = (by 3.1)

a*b + a*ab* = (by 4.1)

(ε+aa*)b + a*ab* = (by 3.1)

εb + aa*b + a*ab* = (by 2.2)

b + aa*b + a*ab* = (by 5.6)

b + aa*b + aa*b* = (by 3.1)

b + aa*(b+b*) = (by 5.2)

b + aa*b*

Use laws to prove equalities

Example: prove that $a^*(b + ab^*) = b + aa^*b^*$.

Proof:

3. Distributive properties
   3.1. $R(S + T) = RS + RT$
   3.2. $(S + T)R = SR + TR$

4.1. $R^* = \varepsilon + RR^* = \varepsilon + R^*R$

$a^*(b+ab^*)$ = (by 3.1)

$a^*b + a^*ab^*$ = (by 4.1)

$(\varepsilon+aa^*)b + a^*ab^*$ = (by 3.1)

$\varepsilon b + aa^*b + a^*ab^*$ = (by 2.2)

$b + aa^*b + a^*ab^*$ = (by 5.6)

$b + aa^*b + aa^*b^*$ = (by 3.1)

$b + aa^*(b+b^*)$ = (by 5.2)

$b + aa^*b^*$

Use laws to prove equalities

Example: prove that a*(b + ab*) = b + aa*b*.

Proof:

a*(b+ab*) = (by 3.1)

a*b + a*ab* = (by 4.1)

(ε+aa*)b + a*ab* = (by 3.1)

εb + aa*b + a*ab* = (by 2.2)

b + aa*b + a*ab* = (by 5.6)

b + aa*b + aa*b* = (by 3.1)

b + aa*(b+b*) = (by 5.2)

b + aa*b*

3. Distributive properties
   3.1. R(S + T) = RS + RT
   3.2. (S + T)R = SR + TR

4.1. R* = ε + RR* = ε + R*R

2.2. Rε = εR = R
2.3. (RS)T = R(ST)

Use laws to prove equalities

Example: prove that $a^*(b + ab^*) = b + aa^*b^*$.

Proof:

$a^*(b+ab^*)$ = (by 3.1)

$a^*b + a^*ab^*$ = (by 4.1)

$(\varepsilon+aa^*)b + a^*ab^*$ = (by 3.1)

$\varepsilon b + aa^*b + a^*ab^*$ = (by 2.2)

$b + aa^*b + a^*ab^*$ = (by 5.6)

$b + aa^*b + aa^*b^*$ = (by 3.1)

$b + aa^*(b+b^*)$ = (by 5.2)

$b + aa^*b^*$

3. Distributive properties
   - 3.1. $R(S + T) = RS + RT$
   - 3.2. $(S + T)R = SR + TR$

4.1. $R^* = \varepsilon + RR^* = \varepsilon + R^*R$

2.2. $R\varepsilon = \varepsilon R = R$

2.3. $(RS)T = R(ST)$

5.6. $R^*R = RR^*$

Use laws to prove equalities

Example: prove that $a^*(b + ab^*) = b + aa^*b^*$.

Proof:

$a^*(b+ab^*)$ = (by 3.1)

3. Distributive properties
   3.1. $R(S + T) = RS + RT$
   3.2. $(S + T)R = SR + TR$

$a^*b + a^*ab^*$ = (by 4.1)

4.1. $R^* = \varepsilon + RR^* = \varepsilon + R^*R$

$(\varepsilon+aa^*)b + a^*ab^*$ = (by 3.1)

$\varepsilon b + aa^*b + a^*ab^*$ = (by 2.2)

2.2. $R\varepsilon = \varepsilon R = R$
2.3. $(RS)T = R(ST)$

$b + aa^*b + a^*ab^*$ = (by 5.6)

5.6. $R^*R = RR^*$

$b + aa^*b + aa^*b^*$ = (by 3.1)

$b + aa^*(b+b^*)$ = (by 5.2)

5.2. $R^* = R^*R^* = (R^*)^* = R + R^*$

$b + aa^*b^*$