COM2109

Automata

Robert Hierons

Regular Expressions

Regular Expressions

Instead of an **automaton**, one can use a RE to describe a regular language

Example: $(a+b\cdot c)*$ = also written: $(a+b\circ c)*$

$(a+bc)*$

describes the language

$$\{a,bc\}*=\{e,a,bc,aa,abc,bca,\ldots\}$$

Recursive Definition

Primitive regular expressions: Æ, *e*, *a*

Given regular expressions r_1 and r_2

$$\left.\begin{array}{l} r_1 + r_2 \xrightarrow{\text{also written as}} r_1 \cup r_2 \\ r_1 \cdot r_2 \\ r_1 * \\ (r_1) \end{array}\right\} \text{Are regular expressions}$$

Examples

A regular expression: $(a+b\cdot c)^*\cdot(c+\varnothing)$

Not a regular expression
(syntactically incorrect): $(a+b+)$

Other regular expressions:

Σ^* Σ^*1 $(0\Sigma^*)\cup(\Sigma^*1)$

$(a\cup b\cdot c)^*$ $\Sigma\Sigma^*1 = \Sigma^+1$

1 or more times

Examples

Precedence rules, line in arithmetic:

2*3+4 is not the same as 2*(3+4)

In REs, unless we have (), precedence:

$^+$ * . + (union)

Examples:

(a+bc)*

(a+b)c*

a+ba*

Why do we care about RE?

- Programs involving **text**:
  - Search for strings that follow certain patterns, e.g. negative words in product reviews: il-, im-, in-, ir-, non-, un-
  - E.g.: happy vs unhappy
  - Part of modern languages like Perl and Python, plus AWK and GREP in Unix
- **Compilers** for programming languages:
  - Tokens (variable names and constants) may be described by REs, based on which automatic systems can generate a **lexical analyzer** – first step of a compiler

Languages of Regular Expressions

$L(r)$: language of regular expression r

Example

$$L\left((a+b\cdot c)^*\right)=\{\varepsilon, a, bc, aa, abc, bca, \ldots\}$$

Definition

For primitive regular expressions:

$$L(Æ) = Æ$$

$$L(e) = \{e\}$$

$$L(a) = \{a\}$$

Definition (continued)

For regular expressions r_1 and r_2

$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

$$L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$$

$$L(r_1 *) = (L(r_1))*$$

$$L((r_1)) = L(r_1)$$

Example

Regular expression: $(a+b)\cdot a^*$

$$L((a+b)\cdot a^*) = L((a+b))\,L(a^*)$$

$$= L(a+b)\,L(a^*)$$

$$= (L(a)\cup L(b))\,(L(a))^*$$

$$= (\{a\}\,\text{È}\,\{b\})\,(\{a\})^*$$

$$= \{a,b\}\,\{e, a, aa, aaa, \ldots\}$$

$$= \{a, aa, aaa, \ldots, b, ba, baa, \ldots\}$$

Example

Regular expression $r = (a+b)^*(a+bb)$

$$= (\{a\} \cup \{b\})^* (\{a\} \cup \{bb\})$$

$$= (\{a,b\})^* (\{a,bb\})$$

$$L(r) = \{a, bb, aa, abb, ba, bbb, \ldots\}$$

Example

Regular expression $r = (aa)^*(bb)^*b$

$$L(r) = \{a^{2n}b^{2m}b: \quad n, m \geq 0\}$$

Example

Regular expression $r = (0+1)^* 00 (0+1)^*$

$L(r)$ = { all strings containing substring 00 }

Example

Regular expression $r = (1+01)^*(0+e)$

$L(r)$ = { ??? }

Example

Regular expression $r = (1+01)^*(0+e)$

$L(r)$ = { all strings without substring 00 }

Equivalent Regular Expressions

Definition:

Regular expressions r_1 and r_2

are **equivalent** if $L(r_1) = L(r_2)$

Example

L = { all strings without substring 00 }

$$r_1 = (1+01)^*(0+e)$$

$$r_2 = (1^*011^*)^*(0+e)+1^*(0+e)$$

$$L(r_1) = L(r_2) = L$$ ⇒ r_1 and r_2 are equivalent regular expressions

Regular Expressions and Regular Languages (and thus, Finite Automata!)

REs and **DFA/NFA** are equivalent in their description power: RE can be converted into finite automata that recognizes the same (regular) language and vice versa.

Theorem

$$\left\{\begin{array}{l}\text{Languages}\\ \text{Generated by}\\ \text{Regular Expressions}\end{array}\right\} = \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

Proof:

$$\left\{\begin{array}{l}\text{Languages}\\ \text{Generated by}\\ \text{Regular Expressions}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

If a language is described by a RE, then it is regular

$$\left\{\begin{array}{l}\text{Languages}\\ \text{Generated by}\\ \text{Regular Expressions}\end{array}\right\} \supseteq \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

If a language is regular, then it described by a RE

Proof - Part 1

$$\left\{\begin{array}{l}\text{Languages} \\ \text{Generated by} \\ \text{Regular Expressions}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Regular} \\ \text{Languages}\end{array}\right\}$$

If a language is described by a RE, then it is regular

For any regular expression r

the language $L(r)$ is regular

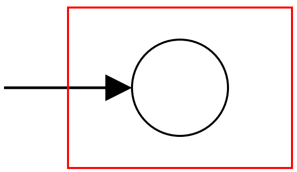
Proof by induction: inductive proof means defining RE in terms of smaller REs

Induction Basis

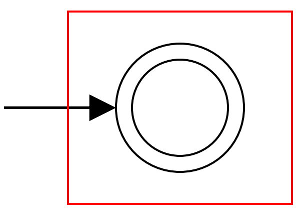
Primitive Regular Expressions: $Æ, e, a$

Corresponding

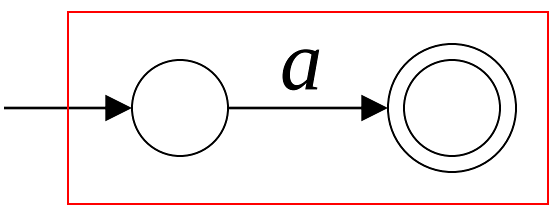
NFAs



$$L(M_1) = \emptyset = L(\emptyset)$$



$$L(M_2) = \{e\} = L(e)$$



$$L(M_3) = \{a\} = L(a)$$

regular languages

Inductive Hypothesis

Suppose

that for regular expressions r_1 and r_2,

$L(r_1)$ and $L(r_2)$ are regular languages

Inductive Step

We will prove:

$$\left.\begin{array}{l} L(r_1 + r_2) \\ \\ L(r_1 \cdot r_2) \\ \\ L(r_1 *) \\ \\ L((r_1)) \end{array}\right\} \text{Are regular Languages}$$

By the definition of regular expressions:

$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

$$L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$$

$$L(r_1 *) = (L(r_1))*$$

$$L((r_1)) = L(r_1)$$

By inductive hypothesis we know:

$L(r_1)$ and $L(r_2)$ are regular languages

We also know:

Regular languages are closed under:

Union	$L(r_1) \cup L(r_2)$
Concatenation	$L(r_1)\,L(r_2)$
Star	$(L(r_1))^*$

Therefore:

$$\left.\begin{array}{l} L(r_1 + r_2) = L(r_1) \cup L(r_2) \\[2em] L(r_1 \cdot r_2) = L(r_1)\, L(r_2) \\[2em] L(r_1 *) = (L(r_1))* \end{array}\right\} \text{Are regular languages}$$

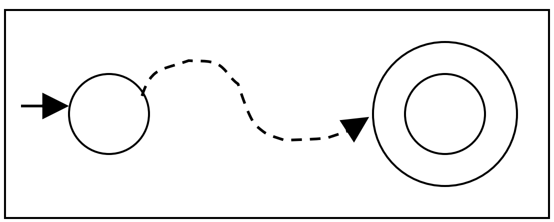
$L((r_1)) = L(r_1)$ is trivially a regular language (by induction hypothesis)

End of Proof-Part 1

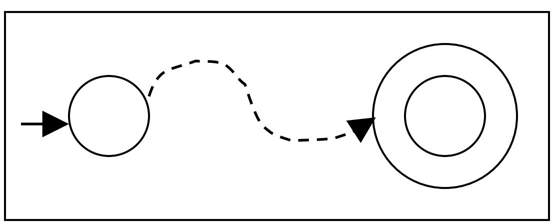
Using the regular closure of these operations, we can construct recursively the NFA M that accepts $L(M) = L(r)$

Example: $r = r_1 + r_2$

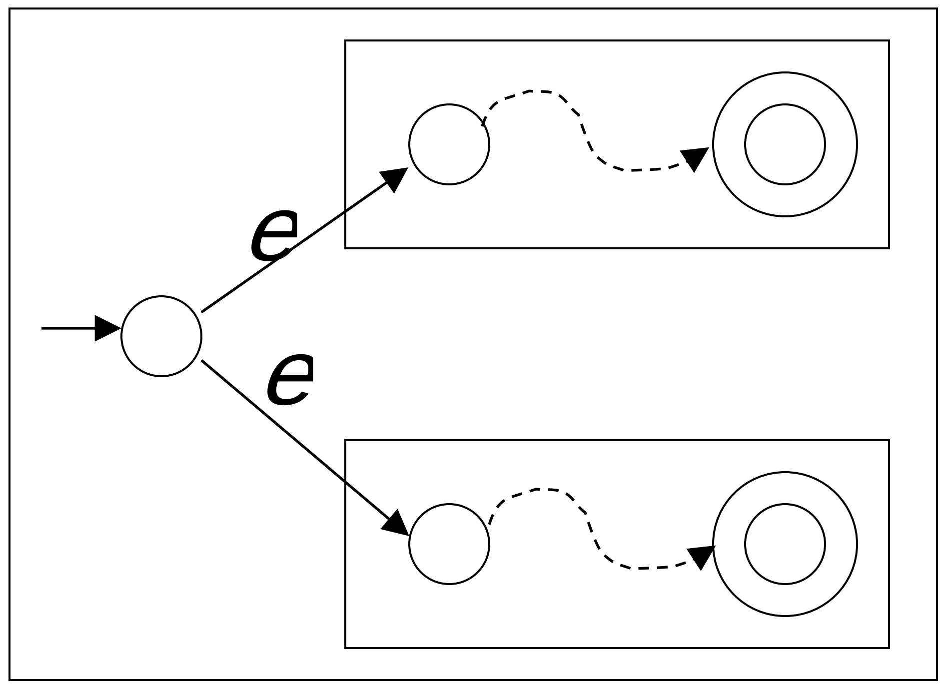
$L(M_1) = L(r_1)$



$L(M_2) = L(r_2)$



$L(M) = L(r)$



Proof - Part 2

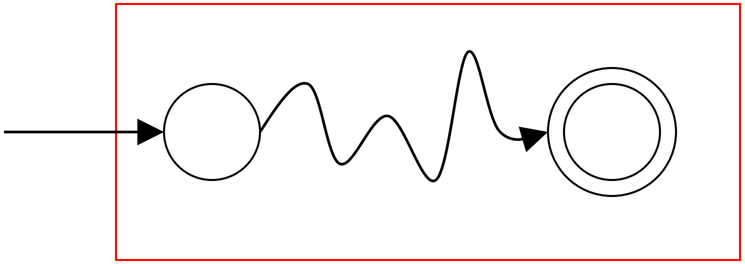
$$\left\{\begin{array}{l}\text{Languages} \\ \text{Generated by} \\ \text{Regular Expressions}\end{array}\right\} \supseteq \left\{\begin{array}{l}\text{Regular} \\ \text{Languages}\end{array}\right\}$$

If a language is regular, then it described by a RE

For any regular language L there is a regular expression r with $L(r) = L$

We will convert an NFA that accepts L to a regular expression

Since L is regular, there is an NFA that accepts it



Can assume it has a single final state (for many initial/final states, it is a '+' of expressions for each start/final pair)

From M construct the equivalent

Generalized Transition Graph

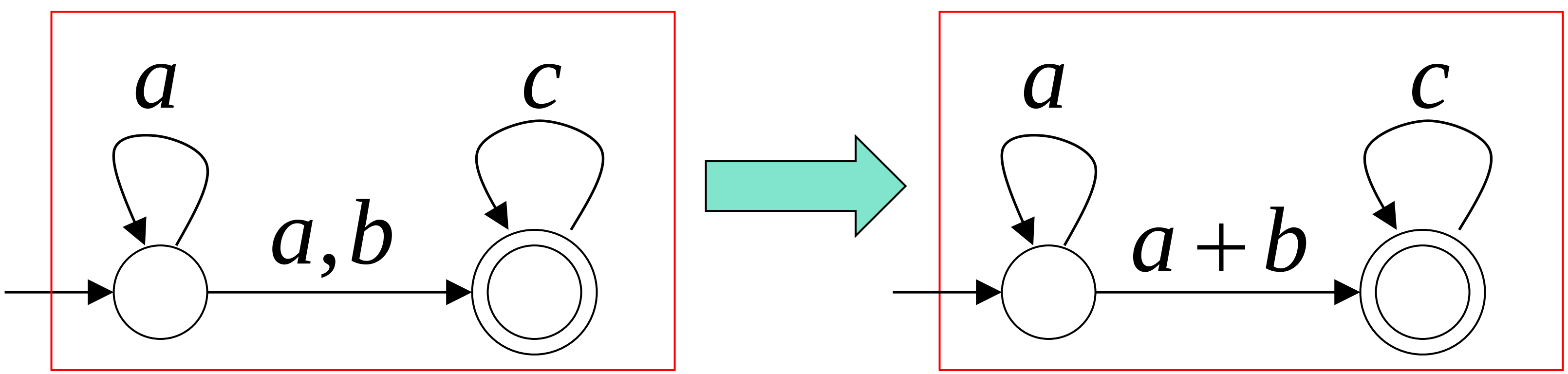
in which transition labels are regular expressions

Example:

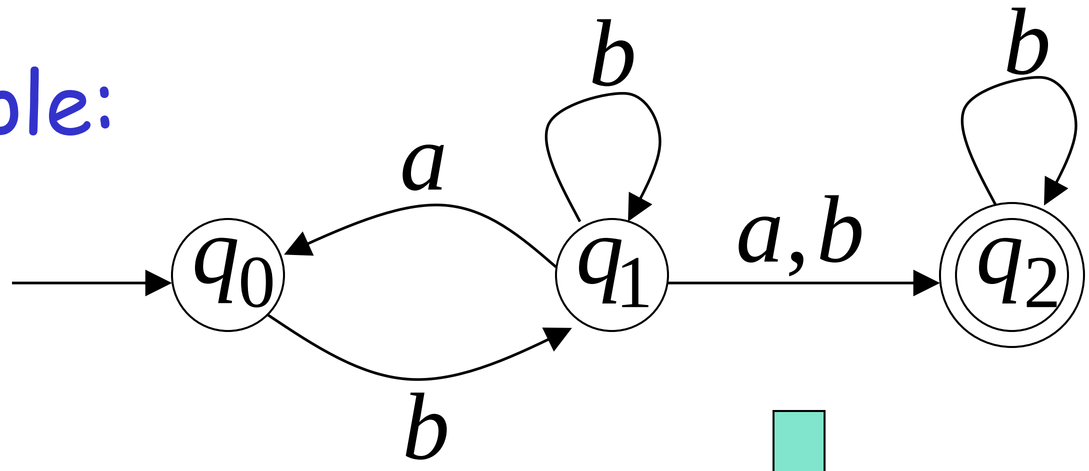
M

Corresponding

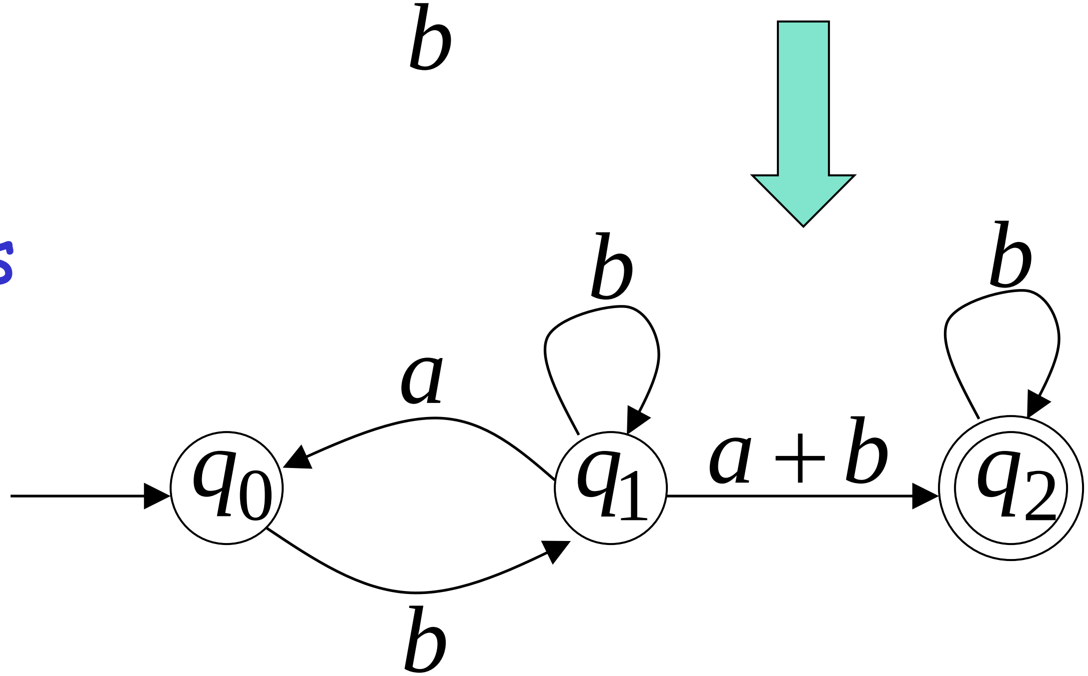
Generalized transition graph



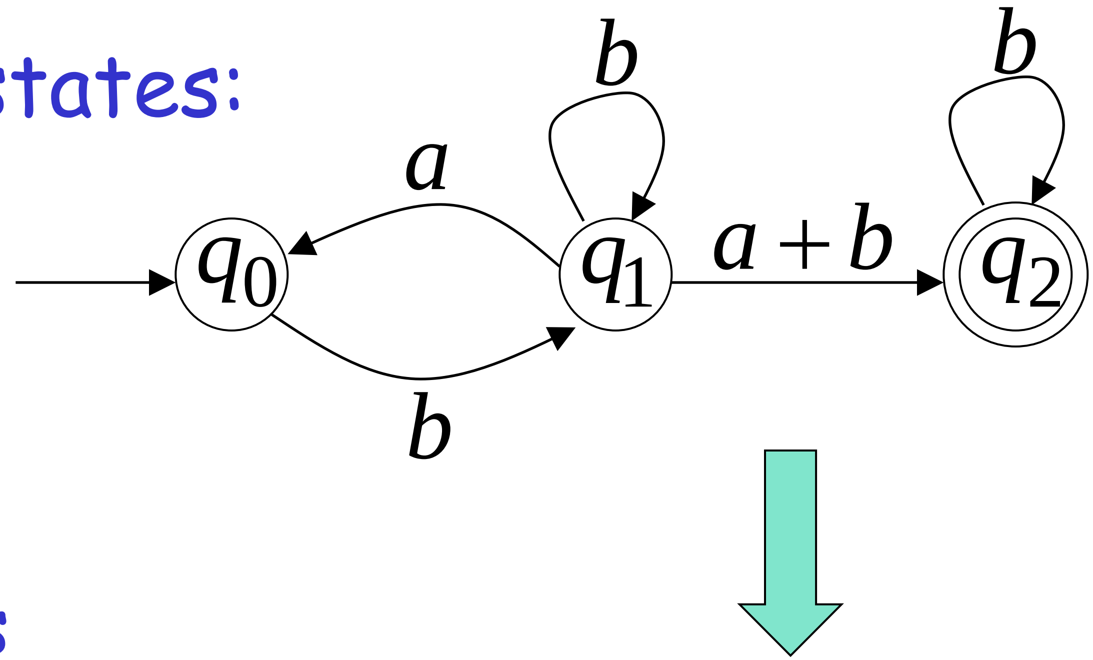
Another Example:



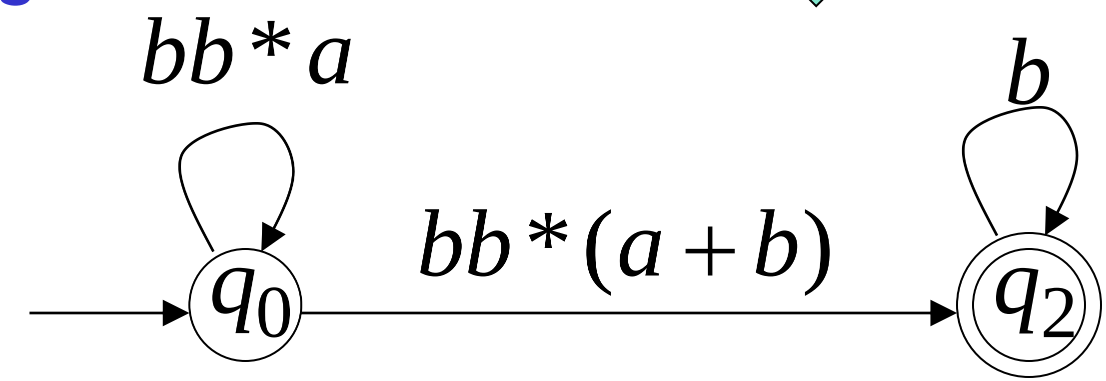
Transition labels are regular expressions



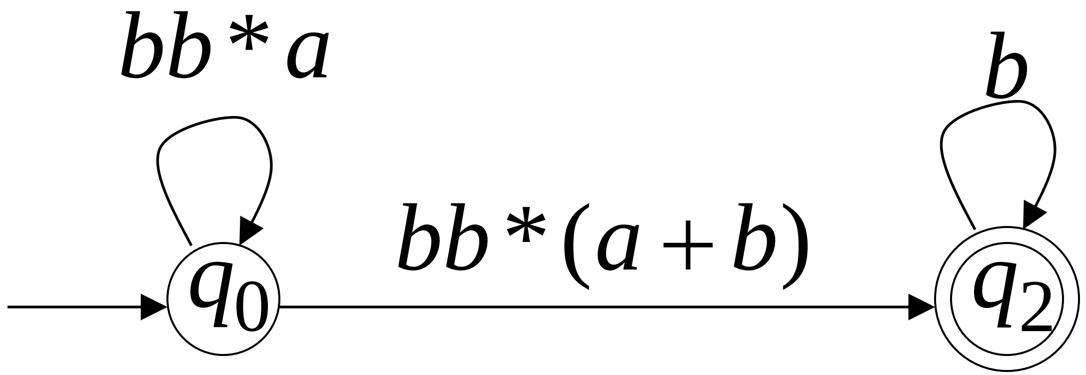
Reducing the states:



Transition labels are regular expressions



Resulting Regular Expression:

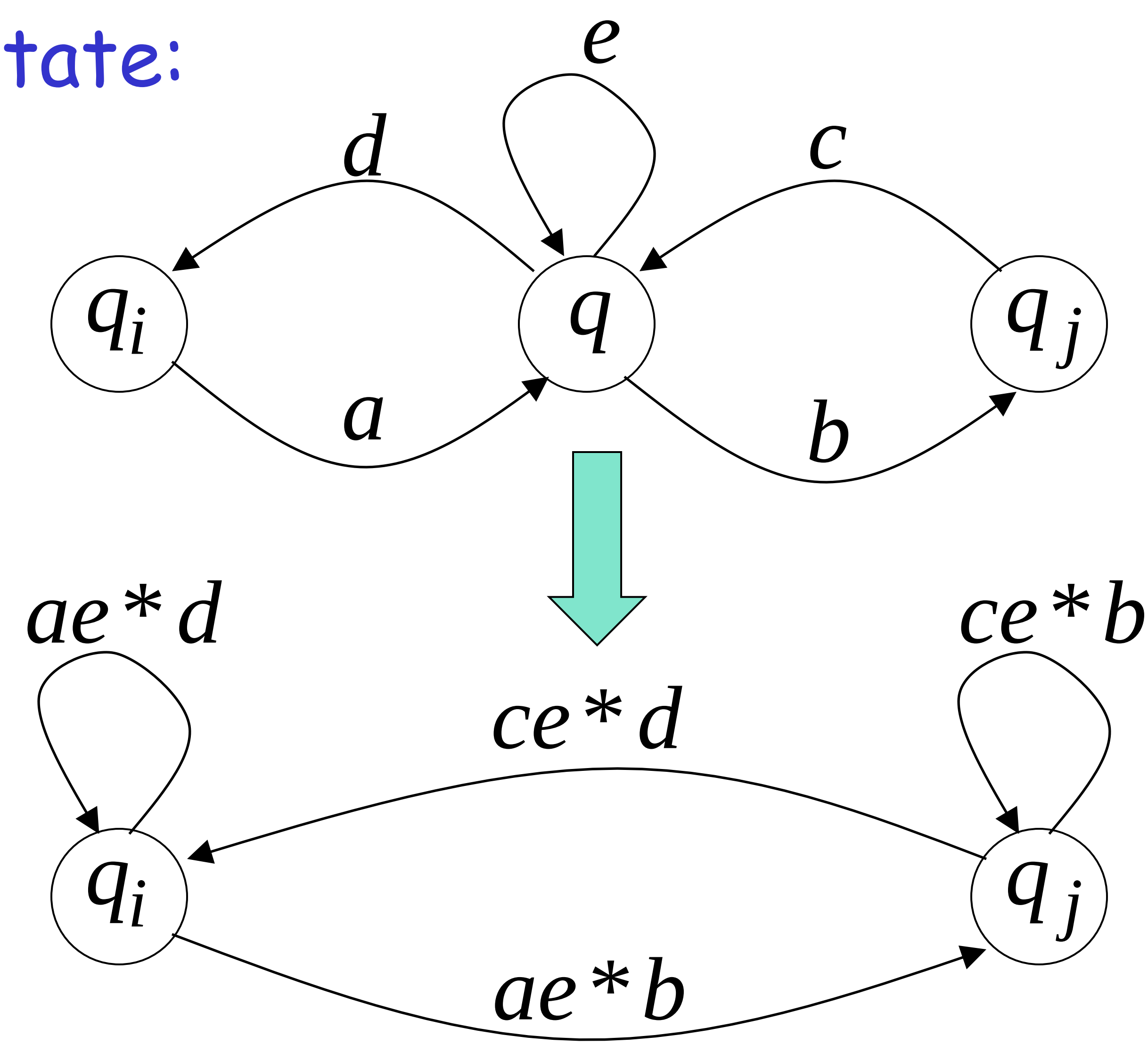


$$r = (bb^*a)^*bb^*(a+b)b^*$$

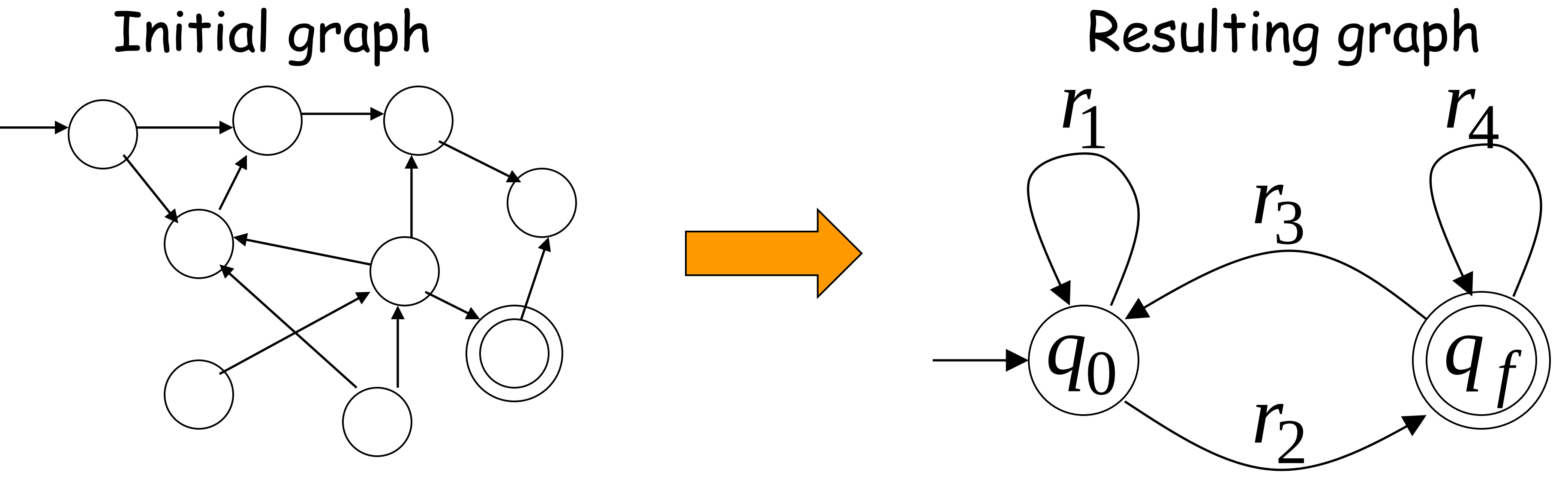
$$L(r) = L(M) = L$$

In General

Removing a state:



By repeating the process until two states are left, the resulting graph is



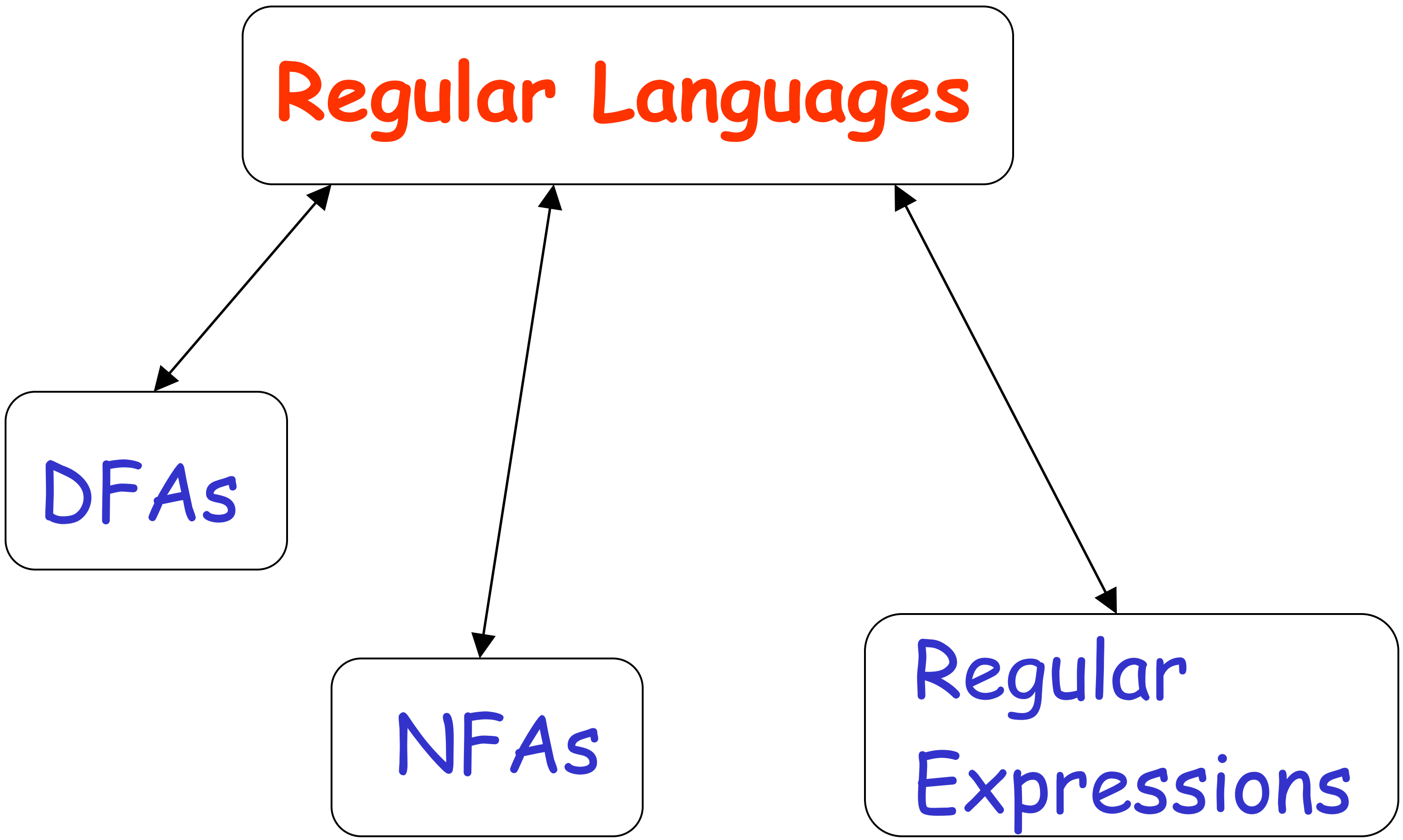
The resulting regular expression:

$$r = r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$$

$$L(r) = L(M) = L$$

End of Proof-Part 2

Standard Representations of Regular Languages



When we say: We are given a Regular Language L

We mean: Language L is in a standard representation

(DFA, NFA, or Regular Expression)