NFA -> DFA, and NFA -> Regexp

Wed, September 23, 2020

HW2

- Working in pairs allowed (but optional)
  - Must notify me who your partner is
  - See new section on course page -> Logistics

- HW1 solutions (partial) will be posted
  - Only after everyone has submitted
  - Volunteers? (contact me)
  - Not ok: submitting someone else’s code
  - Not ok: posting someone else’s code to other websites

- Includes a non-code component
  - Don’t forget about it!

HW1 presentations

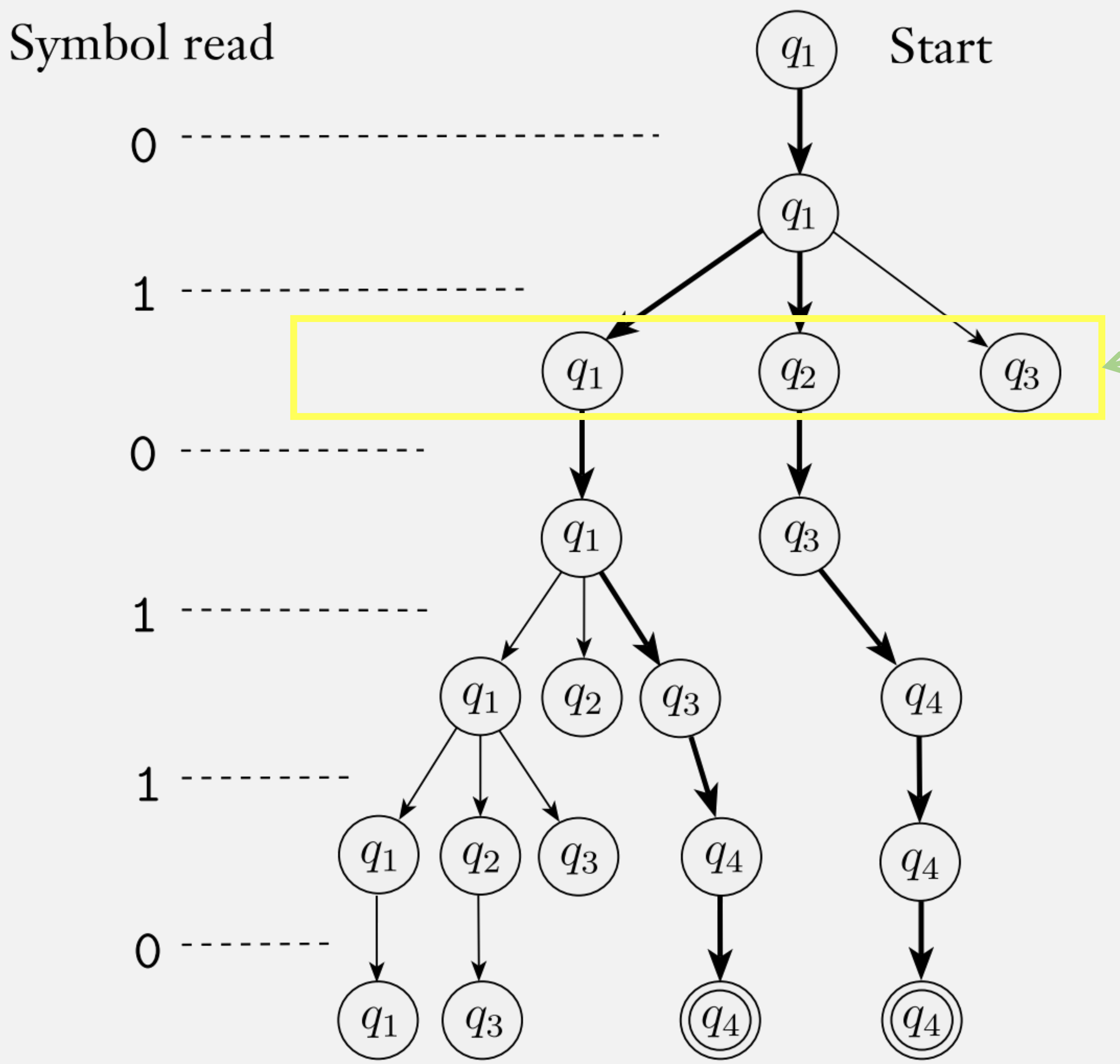
- Paul (Python)
- Laura (Java)
- Nick (Haskell)
- Roy (C++)

See course website for survey forms
(part of your participation grade!)

Proving NFAs recognize regular langs

- *Theorem*:
  - A language A is regular if and only if some NFA N recognizes it.

- Must prove:
  - => If A is regular, then some NFA N recognizes it
    - We know: if A is regular, then a **DFA** recognizes it.
    - Convert DFA to an NFA! (easy)
  - <= If an NFA N recognizes A, then A is regular.
    - Convert NFA to DFA



In a DFA, all these states at each step must be only **one** state

So design a state in the converted DFA to be a **set of NFA states**!

Example:

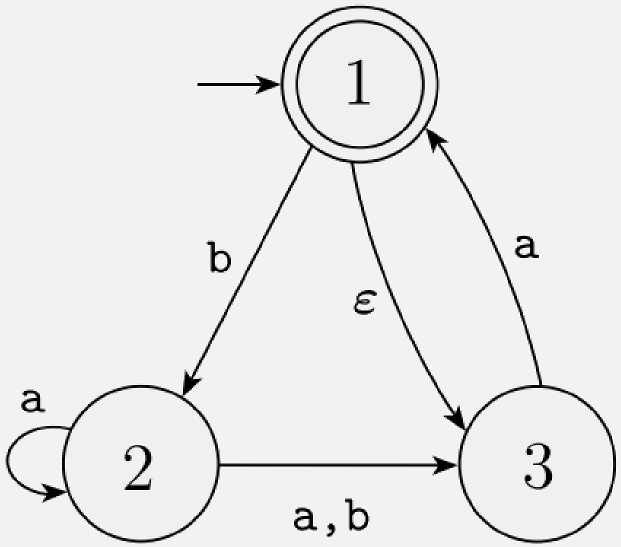


FIGURE 1.42
The NFA N_4

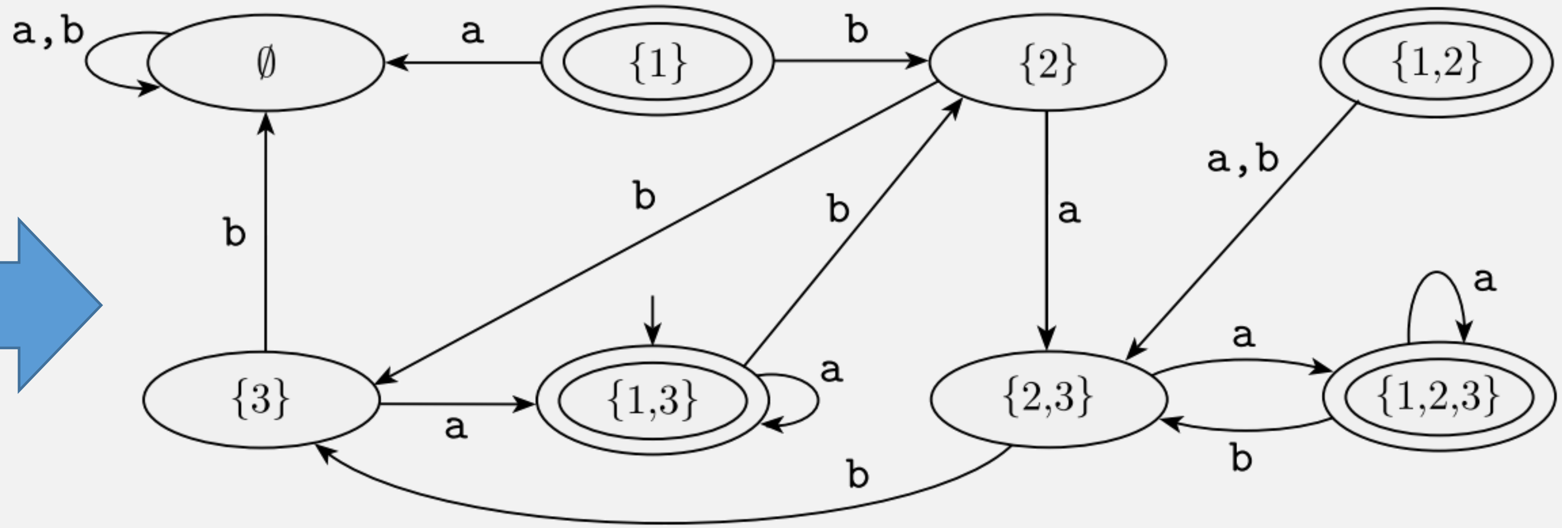


FIGURE 1.43
A DFA D that is equivalent to the NFA N_4

Last time: Convert NFA -> DFA

- Let NFA N = $(Q, \Sigma, \delta, q_0, F)$
- Then equivalent DFA M has states Q' = $\mathcal{P}(Q)$ (power set of Q)

NFA -> DFA (first no empty transitions)

- Have: $N = (Q, \Sigma, \delta, q_0, F)$
- Want: construct a DFA $M = (Q', \Sigma, \delta', {q_0}', F')$

1. $Q' = \mathcal{P}(Q)$.

2. For $R \in Q'$ and $a \in \Sigma$,

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$

For each *r*, "do its transition in *N*", then combine the results into one set

3. ${q_0}' = \{q_0\}$

4. $F' = \{R \in Q' \mid R \text{ contains an accept state of } N\}$

NFA -> DFA (<u>with</u> empty transitions)

- Have: $N = (Q, \Sigma, \delta, q_0, F)$
- Want: construct a DFA $M = (Q', \Sigma, \delta', q_0{}', F')$

1. $Q' = \mathcal{P}(Q)$.

$E(R) = \{q \mid q \text{ can be reached from } R \text{ by traveling along 0 or more } \varepsilon \text{ arrows}\}$

2. For $R \in Q'$ and $a \in \Sigma$,

$$\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a))$$

For each *r*, “do its transition in *N*, then add states reachable from empty transitions”, then combine the results into one set

3. $q_0{}' = E(\{q_0\})$

4. $F' = \{R \in Q' \mid R \text{ contains an accept state of } N\}$

Proving NFAs recognize regular langs

- *Theorem*:
  - A language A is regular if and only if some NFA N recognizes it.

- Must prove:
  - => If A is regular, then some NFA N recognizes it
    - We know: if A is regular, then a **DFA** recognizes it.
    - Convert DFA to an NFA! (easy)
  - <= If an NFA N recognizes A, then A is regular.
    - Convert NFA to DFA, using NFA -> DFA algorithm we just created!

■

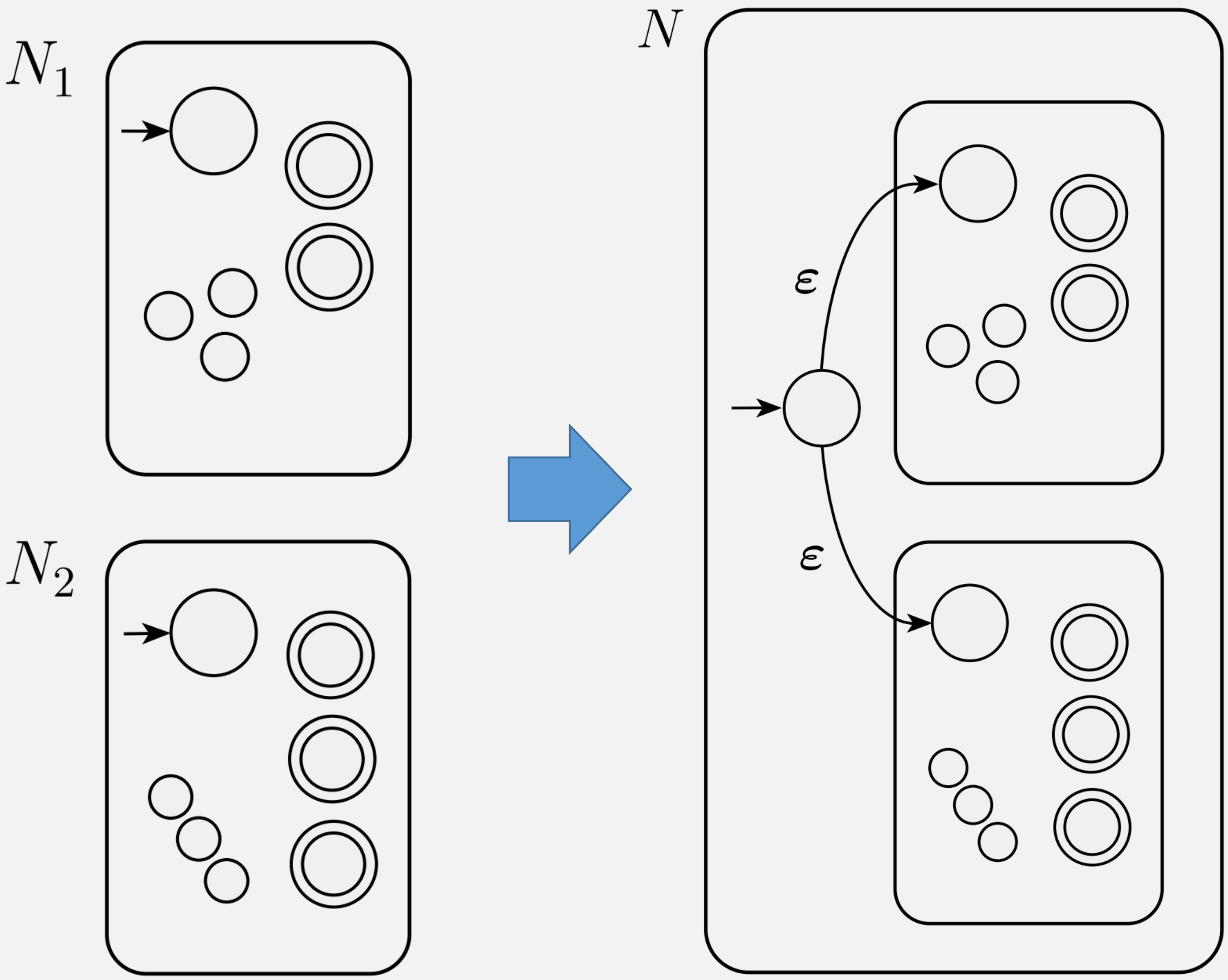
(Q.E.D.)

Regular Operations, Revisited

- Regular languages are closed under the following operations:
  - Union
  - Concatenation
  - Kleene Star

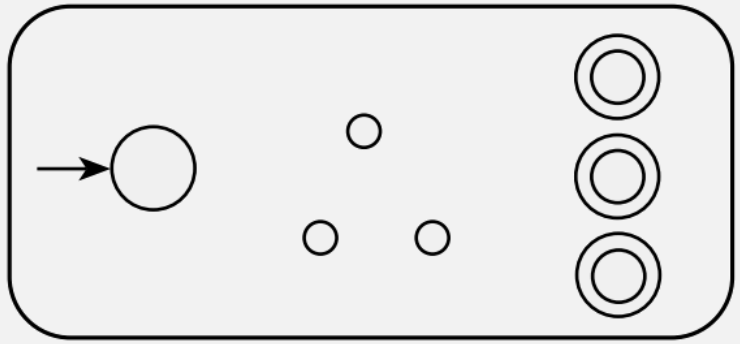
- Easy to prove (by construction) using NFAs

Union

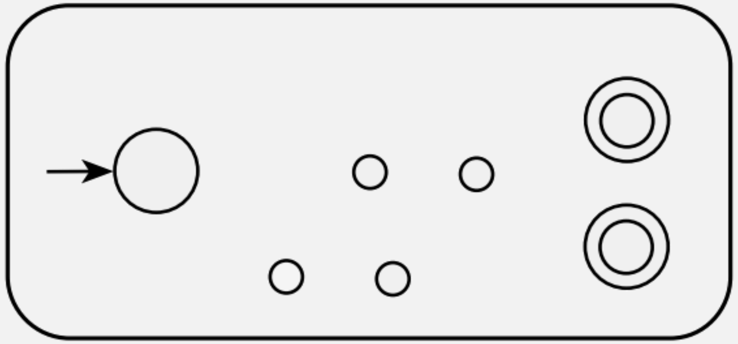


Concat

N_1



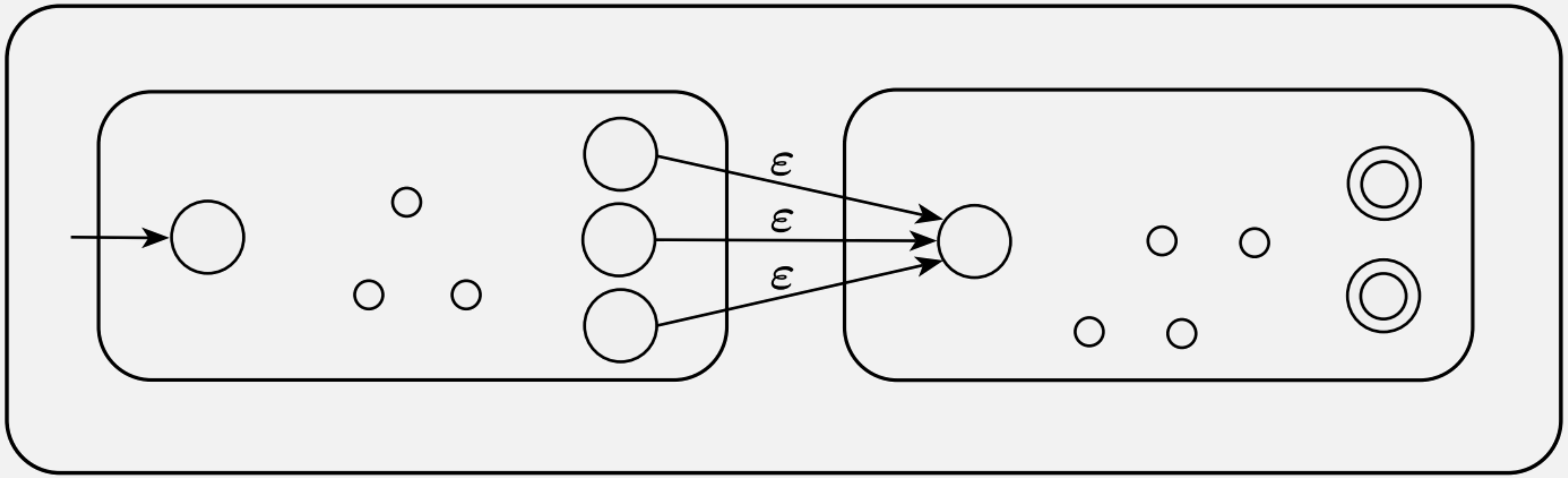
N_2



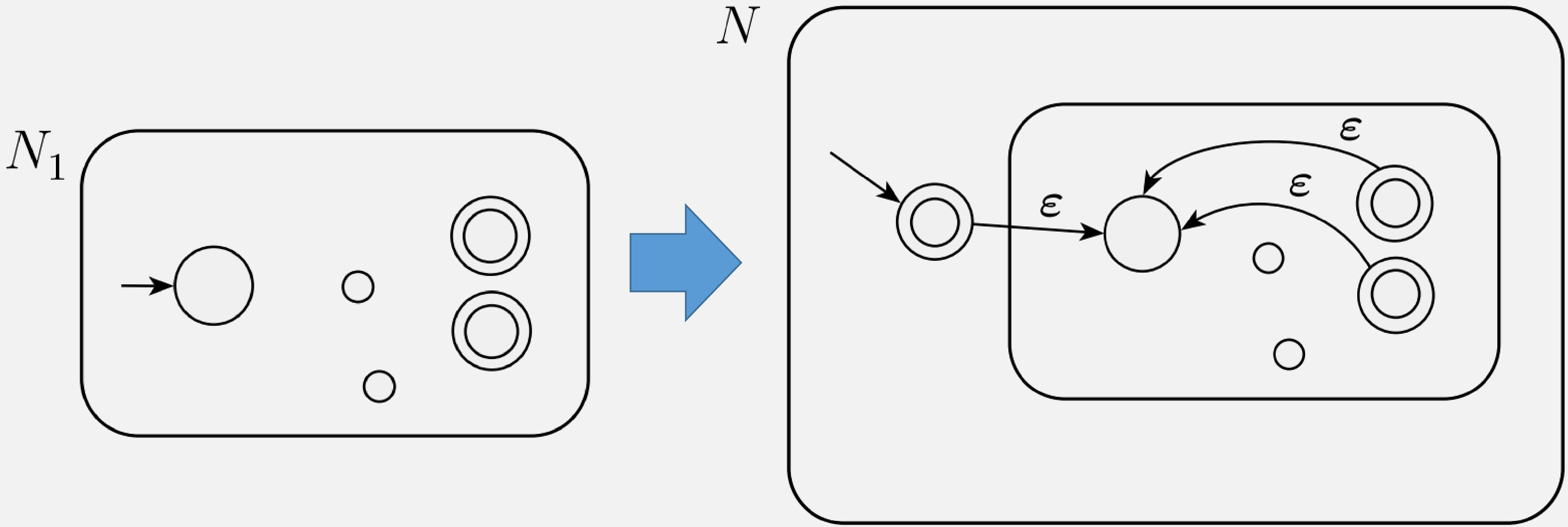
Let N_1 recognize A_1, and N_2 recognize A_2.

Construction of N to recognize $A_1 \circ A_2$

N



Kleene Star



Why do we care?

- Union, concat, and kleene star are sufficient to express all regular languages.
- I.e., they are used to define **regular expressions**

DEFINITION 1.52

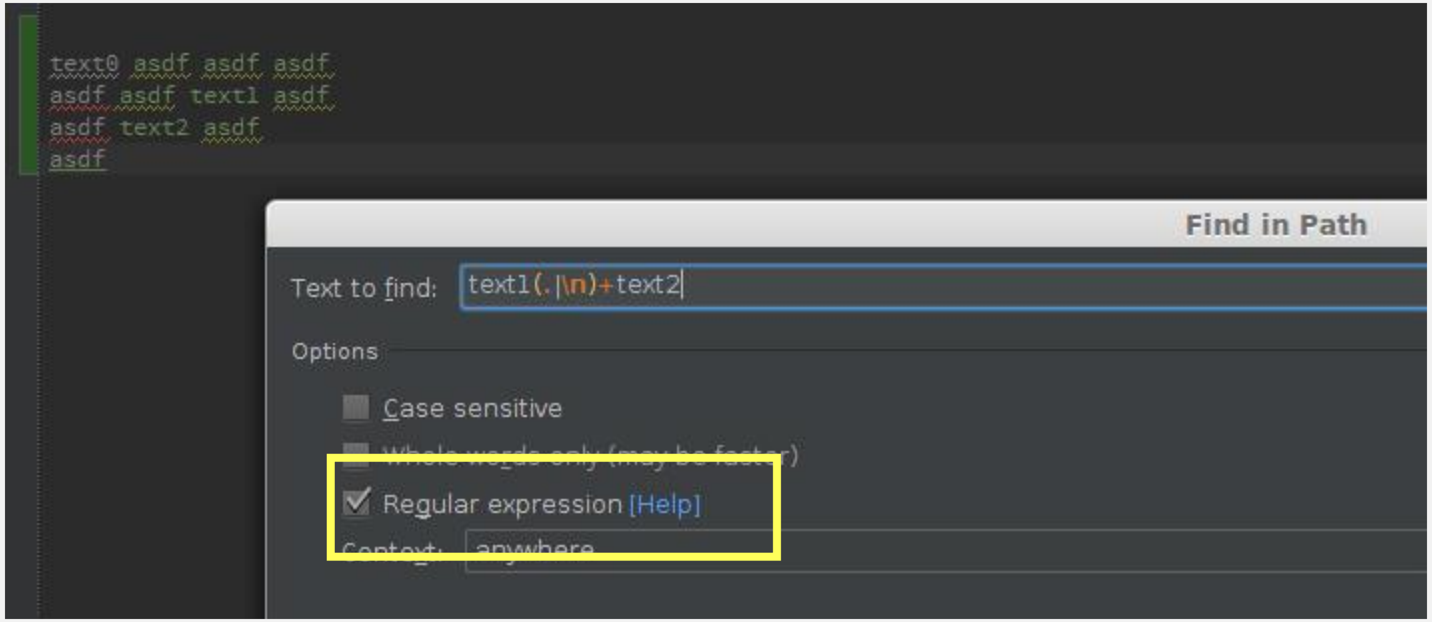
Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
6. (R_1^*), where R_1 is a regular expression.

- E.g., 0*10* = {w | w contains a single 1}

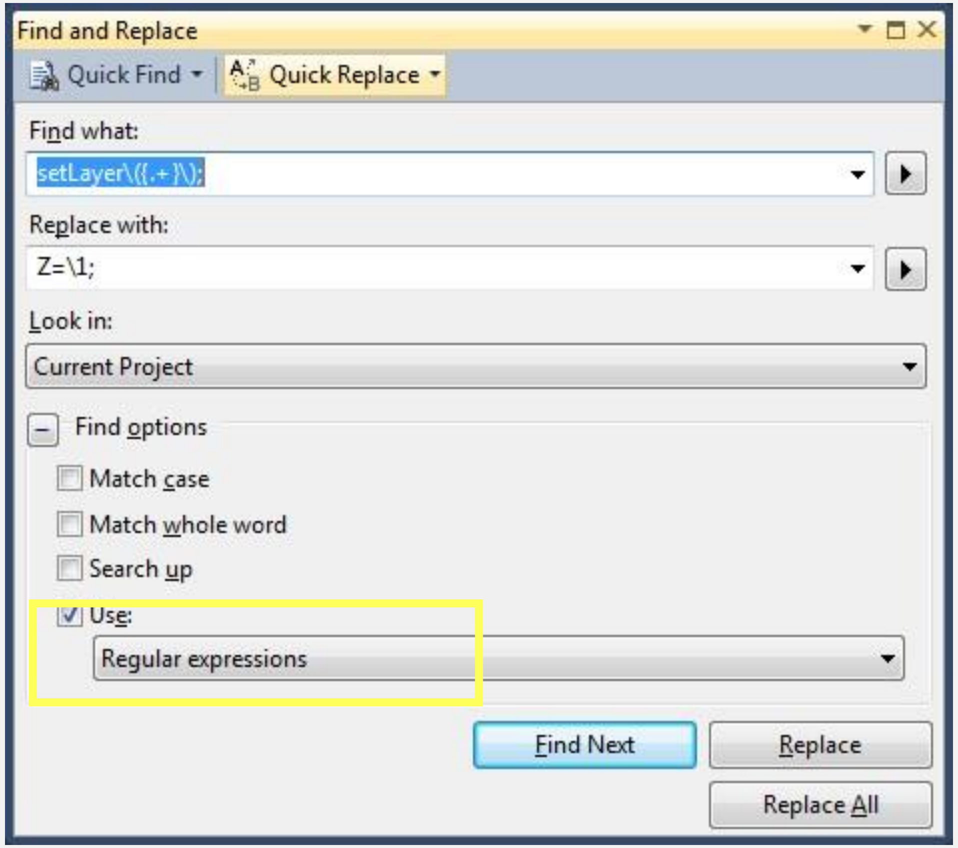
Regular Expressions are Super Useful

- IntelliJ



Regular Expressions are Super Useful

- Visual Studio



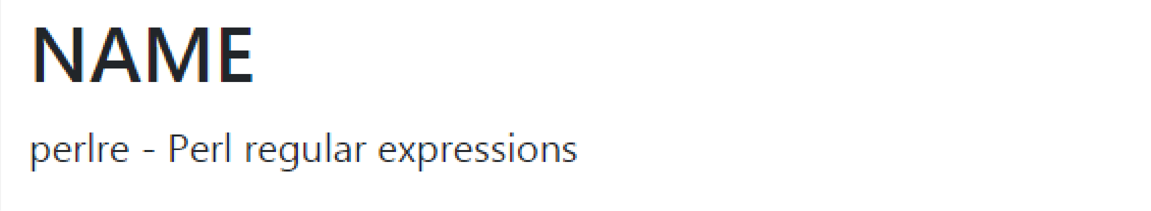
Regular Expressions are Super Useful

- Grep (Linux)

```
GREP(1)                    General Commands Manual                   GREP(1)

NAME
       grep, egrep, fgrep, rgrep - print lines matching a pattern

SYNOPSIS
       grep [OPTIONS] PATTERN [FILE...]
       grep [OPTIONS] [-e PATTERN | -f FILE] [FILE...]

DESCRIPTION
       grep  searches the named input FILEs (or standard input if no files are
       named, or if a single hyphen-minus (-) is given as file name) for lines
       containing  a  match to the given PATTERN.  By default, grep prints the
       matching lines.

       In  addition,  three  variant  programs  egrep,  fgrep  and  rgrep  are
       available.   egrep  is  the  same  as  grep -E.   fgrep  is the same as
       grep -F.  rgrep is the same as grep -r.  Direct  invocation  as  either
       egrep  or  fgrep  is  deprecated,  but  is provided to allow historical
       applications that rely on them to run unmodified.
```

Regexps supported in every language

- Perl
- Python
- Java
- Every lang!



NAME

perlre - Perl regular expressions

Python » English 3.8.6rc1 Documentation » The Python Standard Library » Text Processing Services »

Table of Contents

- re — Regular expression operations
  - Regular Expression Syntax
  - Module Contents
  - Regular Expression

re — Regular expression operations

Source code: Lib/re.py

This module provides regular expression matching operations similar to those found in Perl.

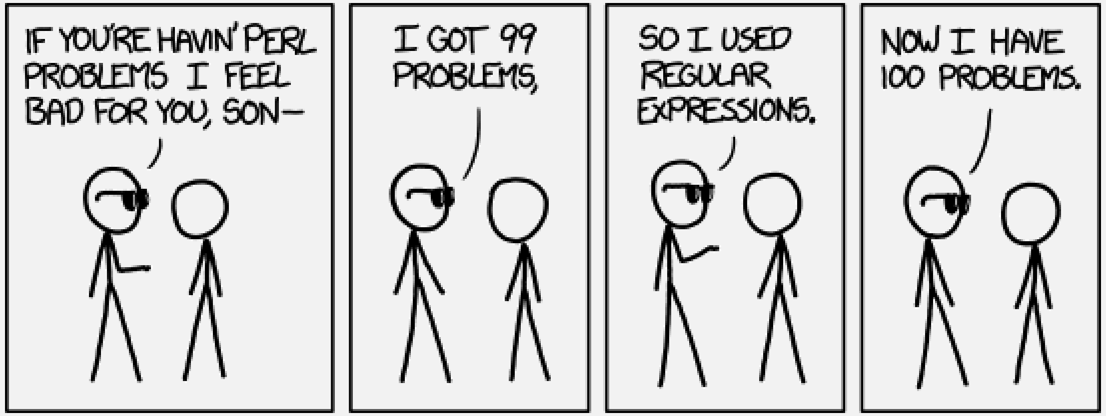
java.util.regex

Class Pattern

java.lang.Object
    java.util.regex.Pattern

Regexps are useful, in the Right Context

Regexps: potentially **useful** …



… but also potentially **bad**

Big Picture Road Map

- We ultimately want to prove:
  - Regular Languages ⇔ Regular Expressions

Next!

- First, we need to show these operations are closed for reglangs:
  - Union (**done**!)
  - Concatentation (**done**!)
  - Kleene star (**done**!)

Thm: A lang is regular iff some regexp describes it

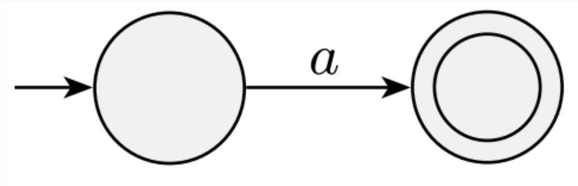
- => If a language is regular, it is described by a regexp

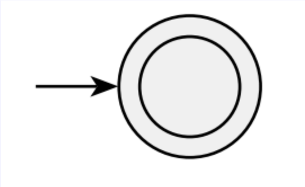
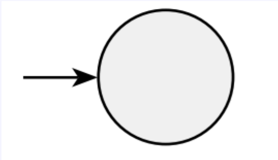
- <= If a language is described by a regexp, it is regular
  - Easy!
  - Construct the NFA!
  - See Lemma 1.55

Regexp -> NFA

DEFINITION 1.52

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ, 

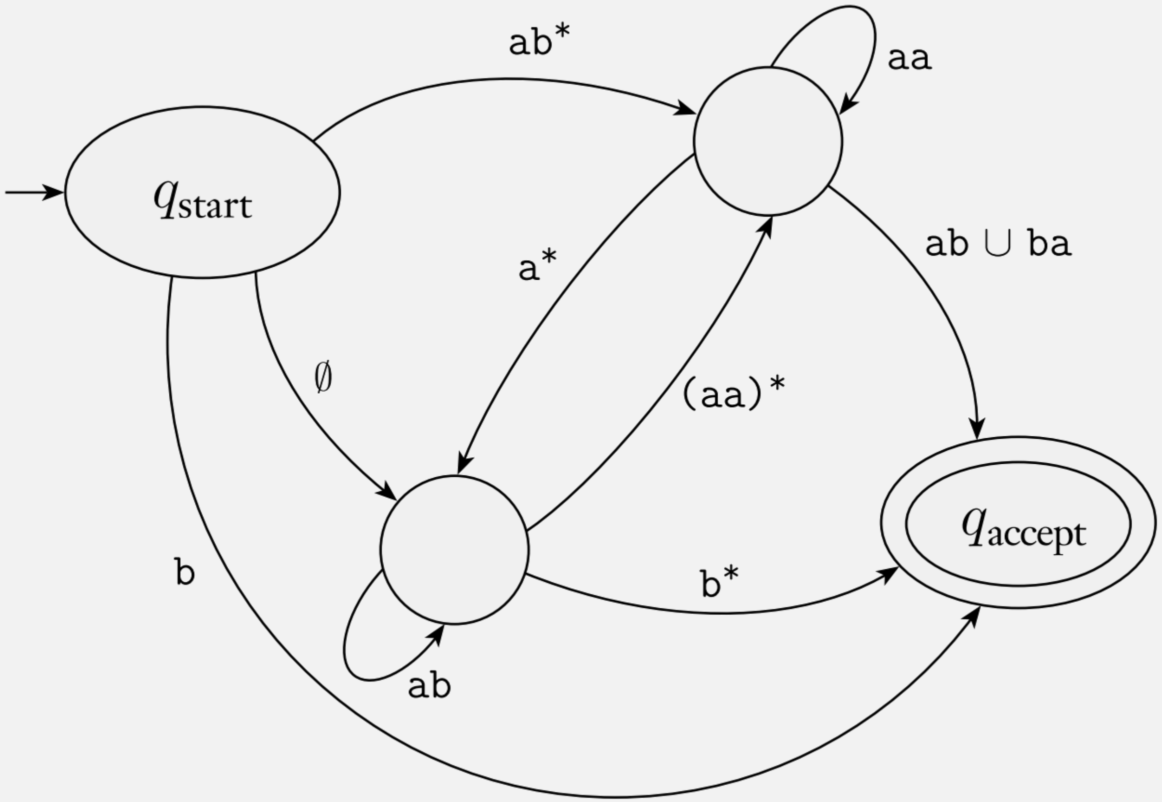
2. ε, 
3. $\emptyset$, 
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R … r
6. (R_1^*), where R_1 is a regular expression.

Constructions from before!

Thm: A lang is regular iff some regexp describes it

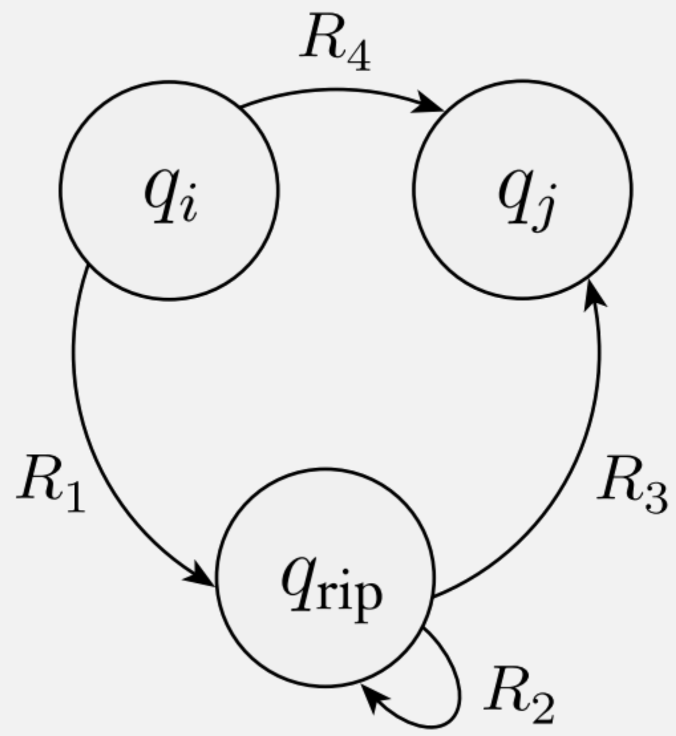
- => If a language is regular, it is described by a regexp
  - Hard!
  - Need something new: a GNFA
- <= If a language is described by a regexp, it is regular
  - Easy!
  - Construct the NFA! (**Done**)

GNFA = NFA with regexp transitions

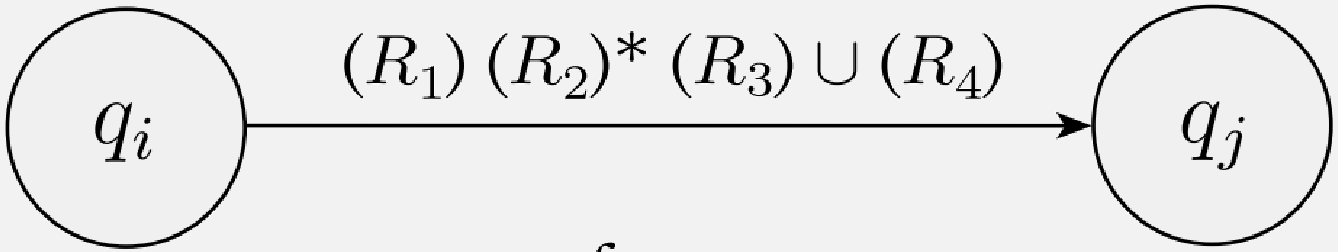


- To <u>convert</u> to regexp, keep "ripping out" states until only 2 are left

CONVERT(G): ripping a state, and patching

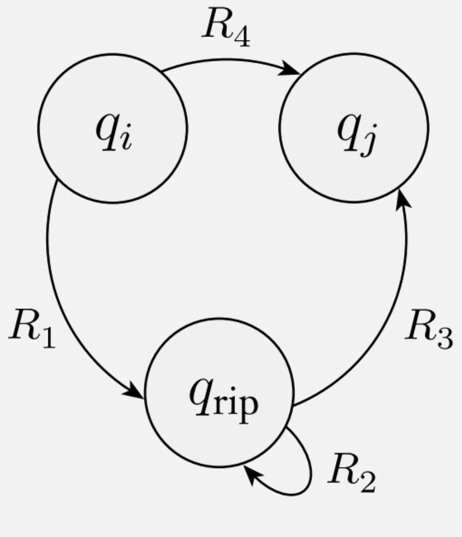


before

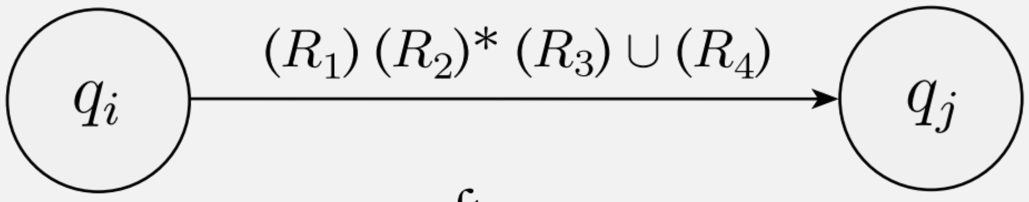


after

Next time: CONVERT(G) function



before



after

- If G has 2 states, then return the regexp
- Else
  - "Rip" out one state to get G'
  - Recursively call CONVERT(G')

Check-in Quiz 9/23

On gradescope

End of Class Survey 9/23

See course website