From exercise class 4:

– Even number of a and even number of b

– Even number of a and even number of b

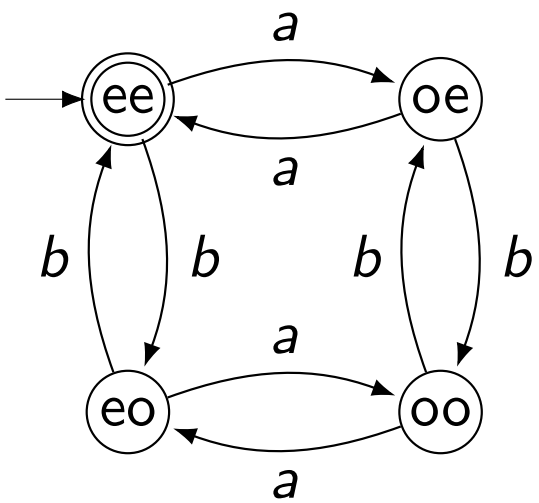
two letters together

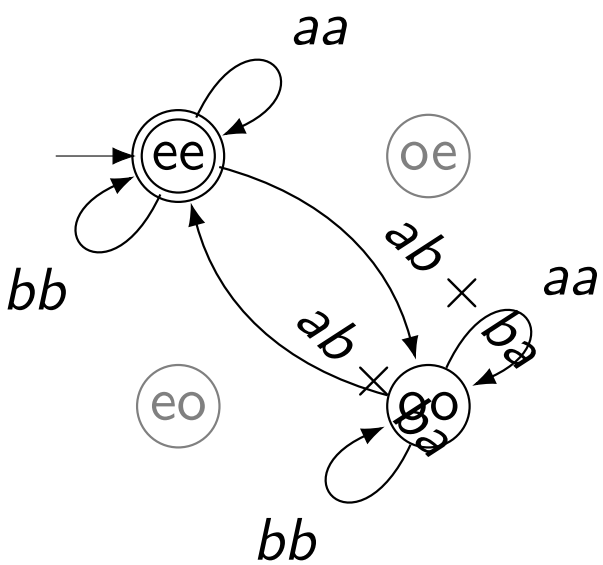
aa and bb keep both numbers even [odd]

ab and ba switch between even and odd, for both numbers

$$(aa + bb + (ab + ba)(aa + bb)^*(ab + ba))^*$$

[M] E 3.4, see ↪Brzozowski *et* McCluskey





– Numeric constants in programming language

14, +1, −12, 14.3, −.99, 16., 3E14, −1.00E2, 4.1E−1, .3E+2

Use d for $(0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9)$

Use s for $(\Lambda + \text{'+'} + \text{'-'})$

Use p for '.'

– Numeric constants in programming language

14, +1, −12, 14.3, −.99, 16., 3E14, −1.00E2, 4.1E−1, .3E+2

Use d for $(0+1+2+3+4+5+6+7+8+9)$

Use s for $(\Lambda + \text{'+'} + \text{'-'})$

Use p for '.'

$$s(dd^*(\Lambda + pd^*) + pdd^*)(\Lambda + Esdd^*)$$

[M] E 3.5

Theorem (Kleene)

Finite automata and regular expressions specify the same family of languages.

- from RegEx to FA
  $\hookrightarrow$Thompson's construction
- from FA to RegEx
  In book: $\hookrightarrow$McNaughton and Yamada
  We do: $\hookrightarrow$Brzozowski et McCluskey

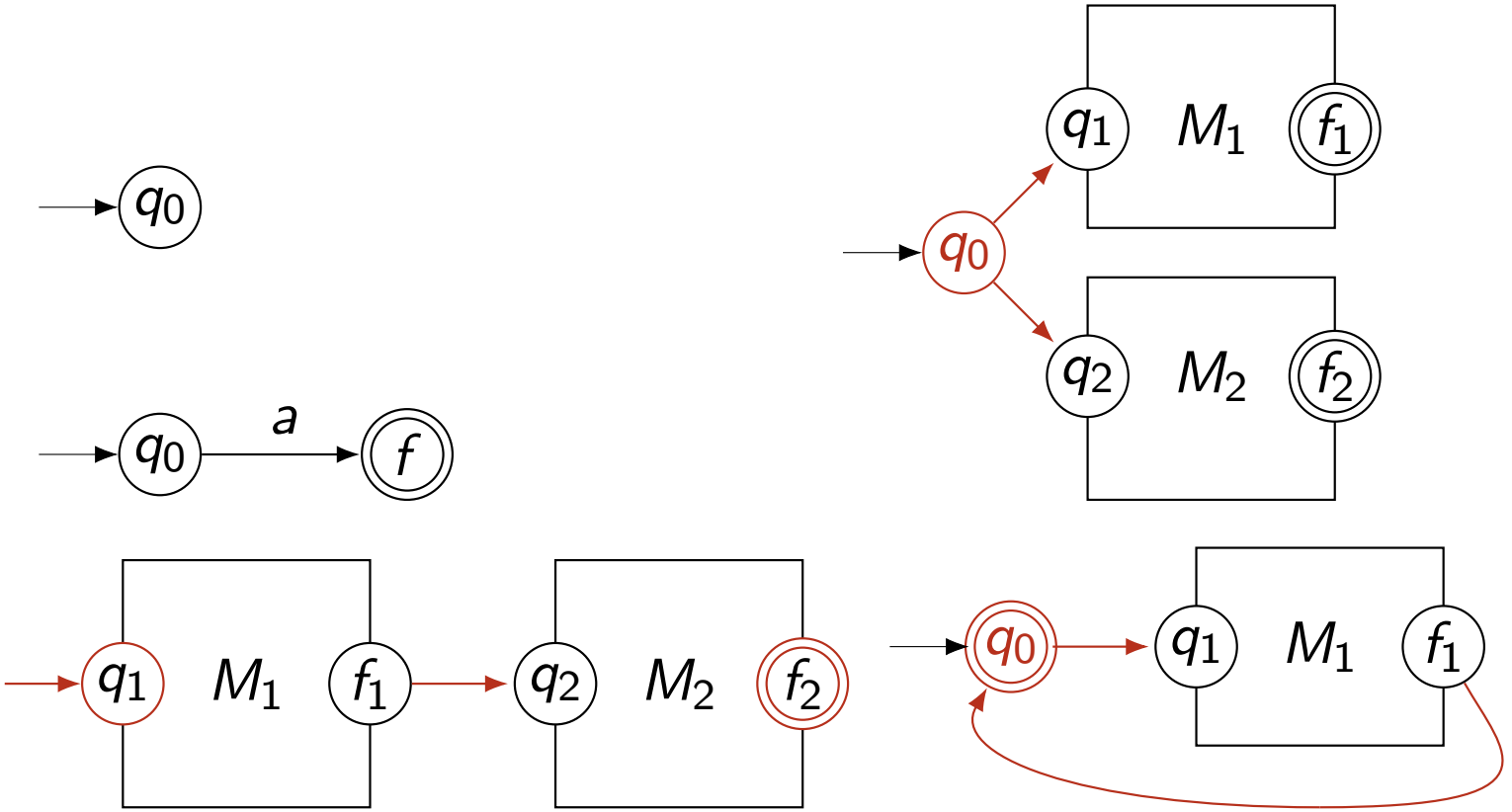
Theorem

If L is a regular language, then there exists an NFA that accepts L.

$\emptyset$
$\{a\}$
$L_1 \cup L_2$
$L_1 \cdot L_2$
L_1^*

Theorem

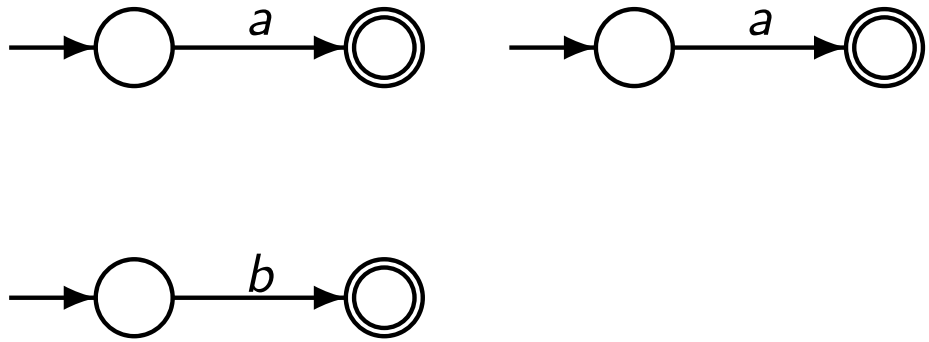
If L is a regular language, then there exists an NFA that accepts L.

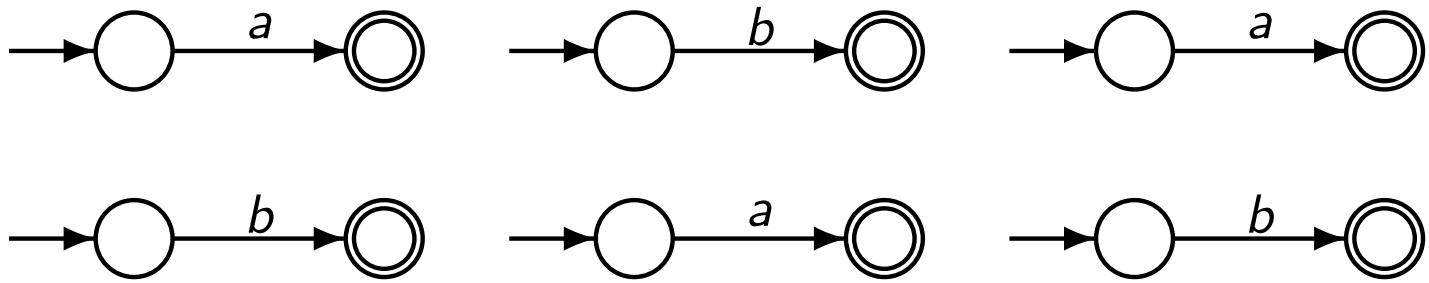


[M] Th 3.25 [L] Th 3.1

Example 3.28. An NFA Corresponding to $((aa + b)^*(aba)^*bab)^*$

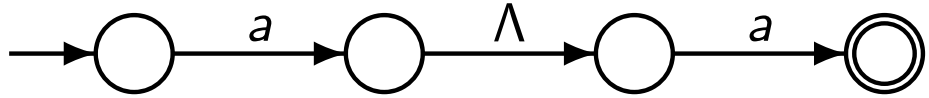
Step 1

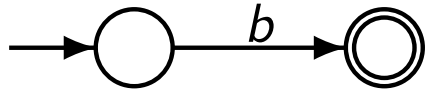


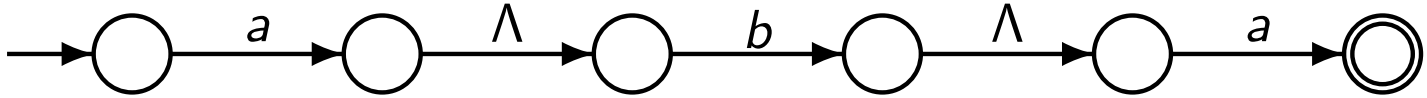


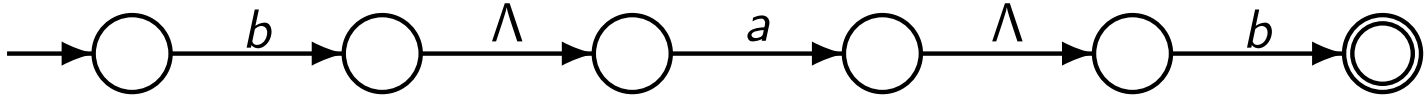
Example 3.28. An NFA Corresponding to $((aa + b)^*(aba)^*bab)^*$

Step 2



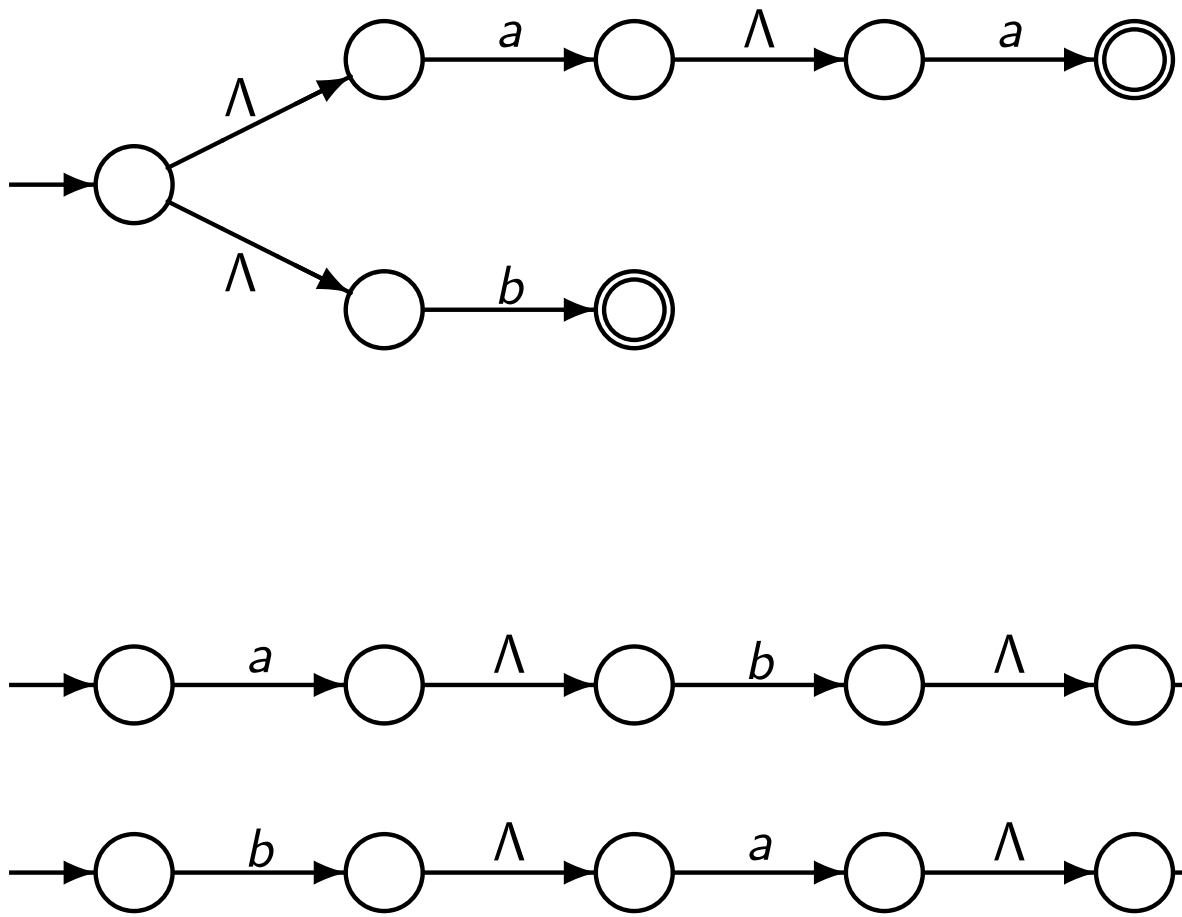






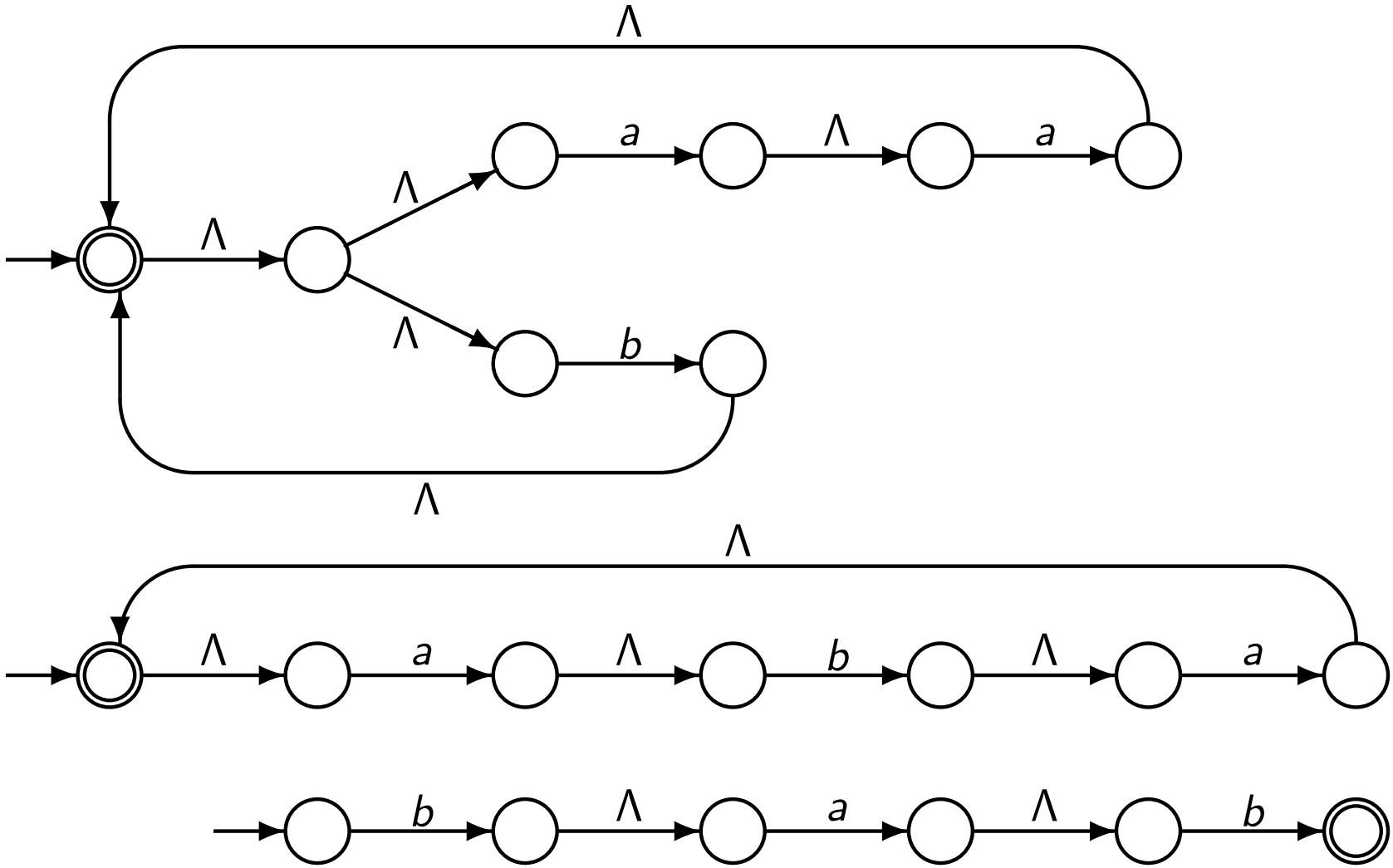
Example 3.28. An NFA Corresponding to $((aa + b)^*(aba)^*bab)^*$

Step 3



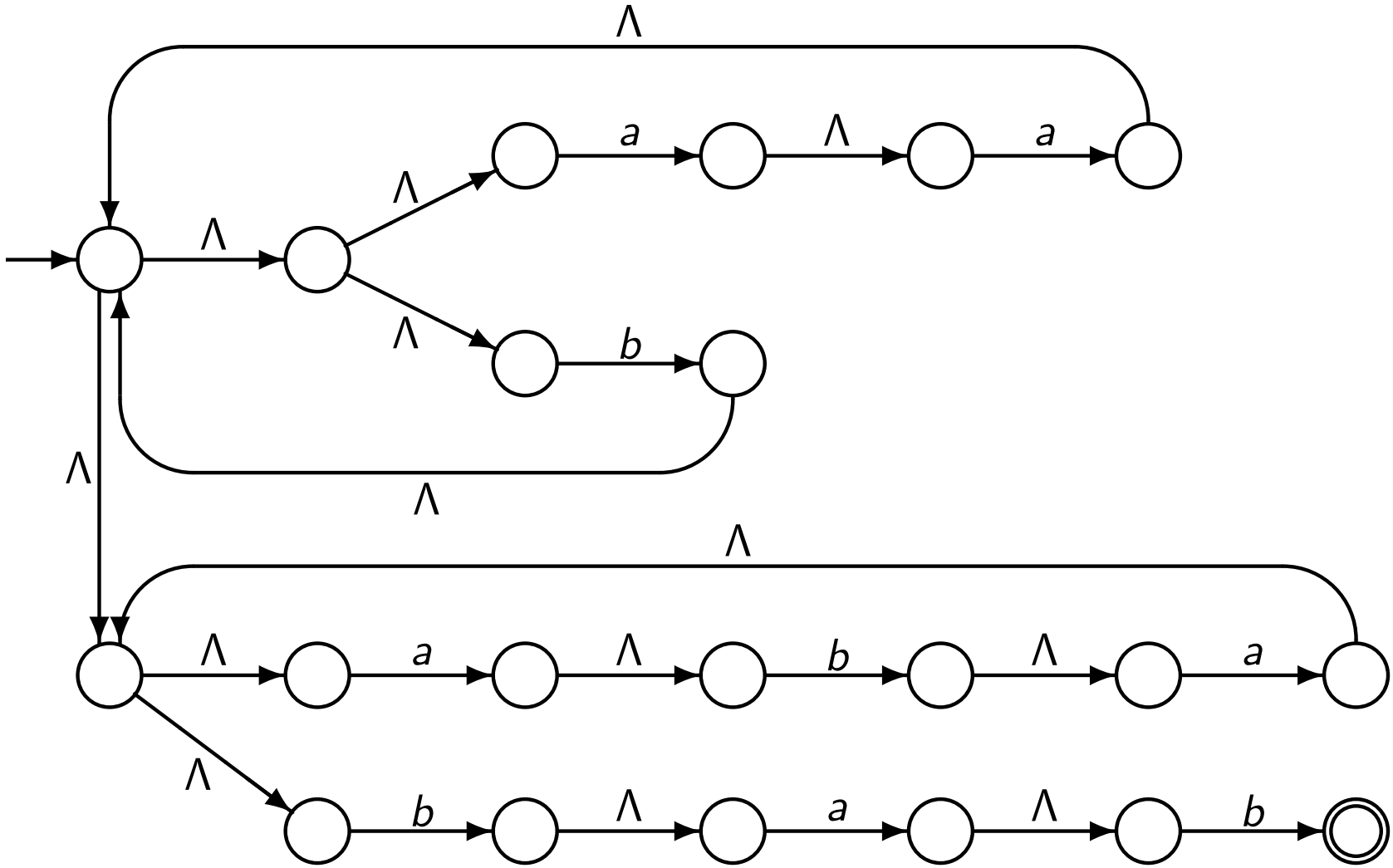
Example 3.28. An NFA Corresponding to $((aa + b)^*(aba)^*bab)^*$

Step 4



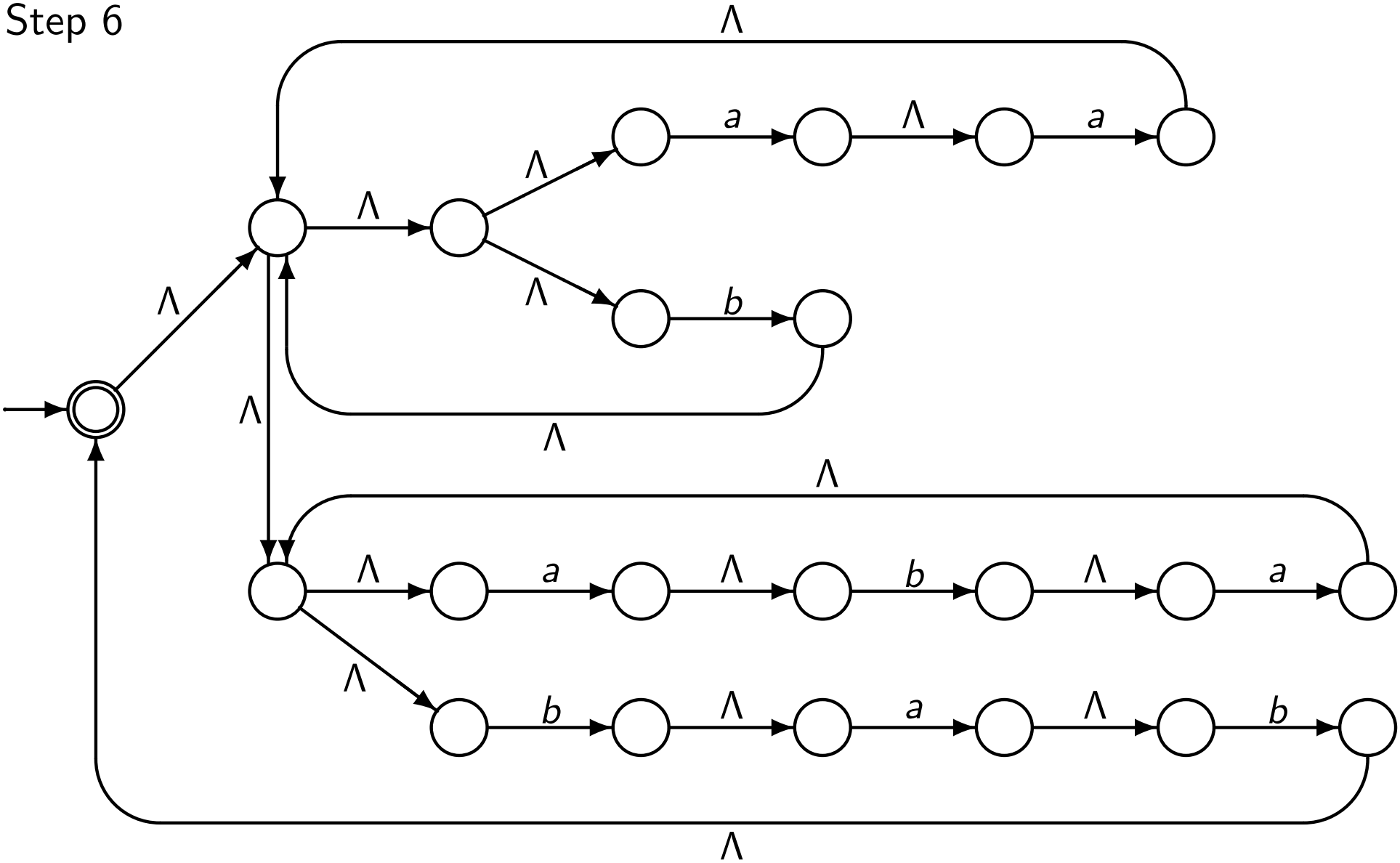
Example 3.28. An NFA Corresponding to $((aa + b)^*(aba)^*bab)^*$

Step 5

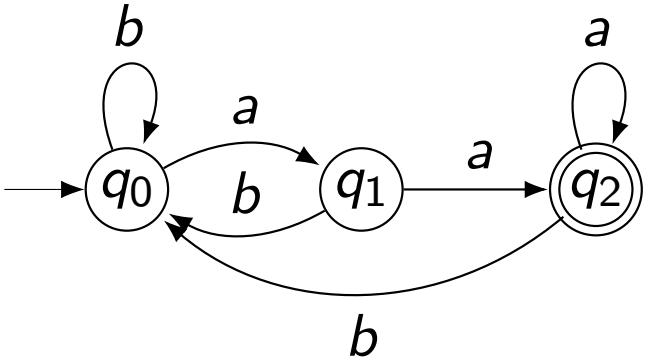


Example 3.28. An NFA Corresponding to $((aa + b)^*(aba)^*bab)^*$

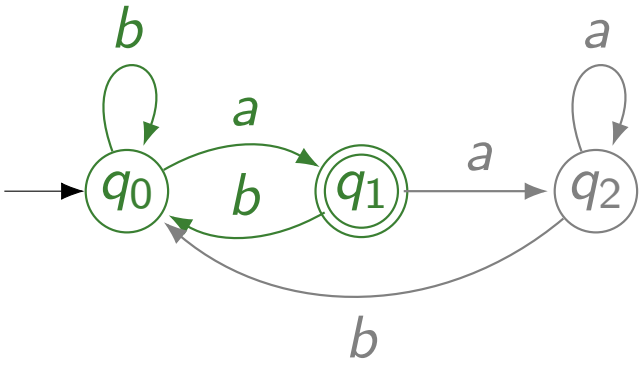
Step 6



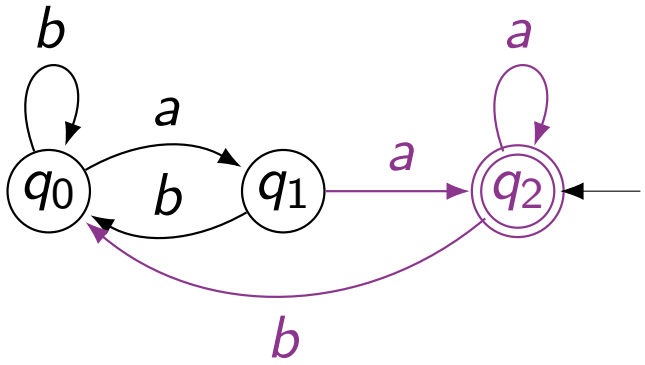
Intro: finding a regular expression



Intro: finding a regular expression



$\underbrace{(b + ab)^* a}_{\text{loop on } q_0}$



$\underbrace{b\,[(b + ab)^* a]\,a + a}_{\text{single loop on } q_2}$

$\underbrace{[(b + ab)^* aa]}_{\text{from } q_0 \text{ to } q_2}\,\underbrace{[b(b + ab)^* aa + a]^*}_{\text{loop on } q_2}$

short answer $(a + b)^* aa$ see $\hookrightarrow$FA example

Note

The *state elimination method* by Brzozowski et McCluskey constructs a regular expression for a given automaton, by iteratively removing the states. The edges of the automaton do not just contain symbols (or Λ) but regular expressions themselves. Thus the graphs are a hybrid form of finite automata and regular expressions. It is rather clear however what they express.

Note (continued)

Start by adding a new initial and accepting state; connect the initial state to the old initial state, and connect the old accepting states to the new accepting state, using as label the expression Λ (representing the empty word).

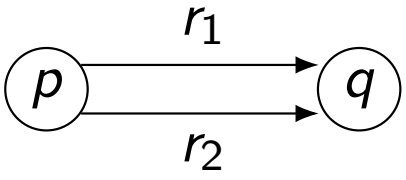
Whenever during this construction two parallel edges (p, r_1, q) and (p, r_2, q) appear, we replace them with a single edge $(p, r_1 + r_2, q)$

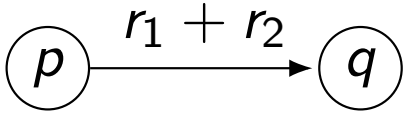
Choose any node q to be removed. Let r_2 be the expression on the loop for q. (If there is no loop we consider this expression to be $\varnothing$.)

For any incoming edge (p, r_1, q) and outgoing edge (q, r_3, s) we add the edge $(p, r_1 r_2^* r_3, s)$ which replace the path from p to s via q.

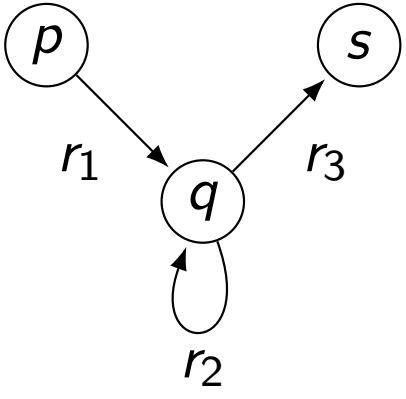
Remove q. Repeat.

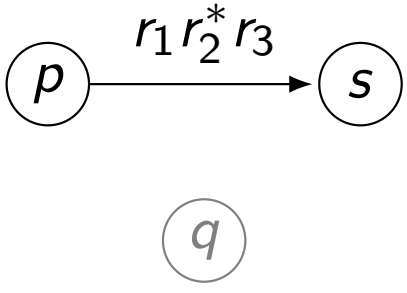
When all original nodes are removed, we obtain a graph with single edge; its label represents the language of the original automaton.



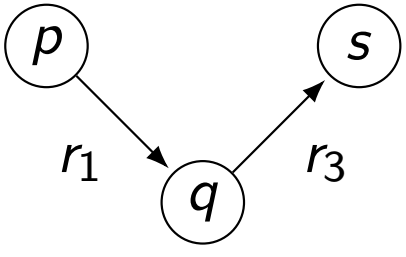


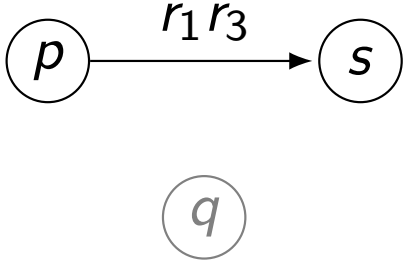
join parallel edges





reduce node q

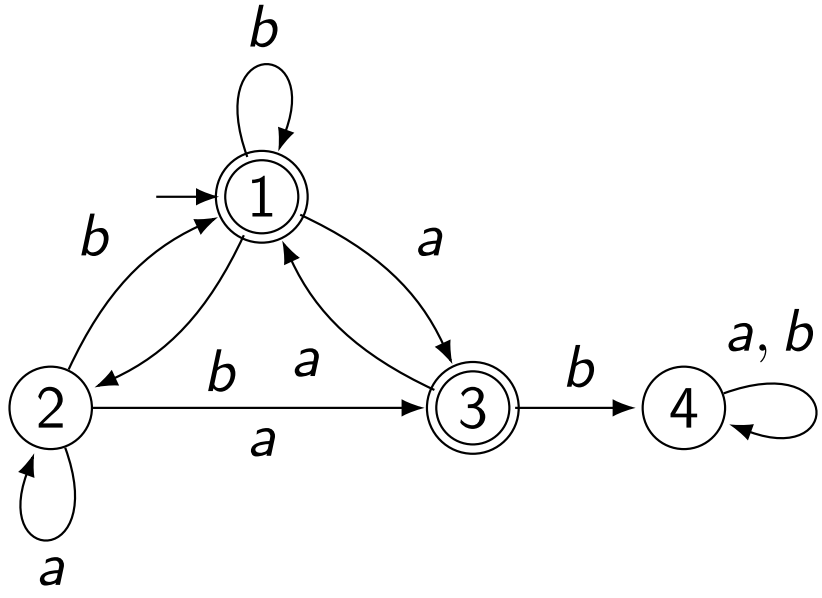




special case: $r_2 = \varnothing$

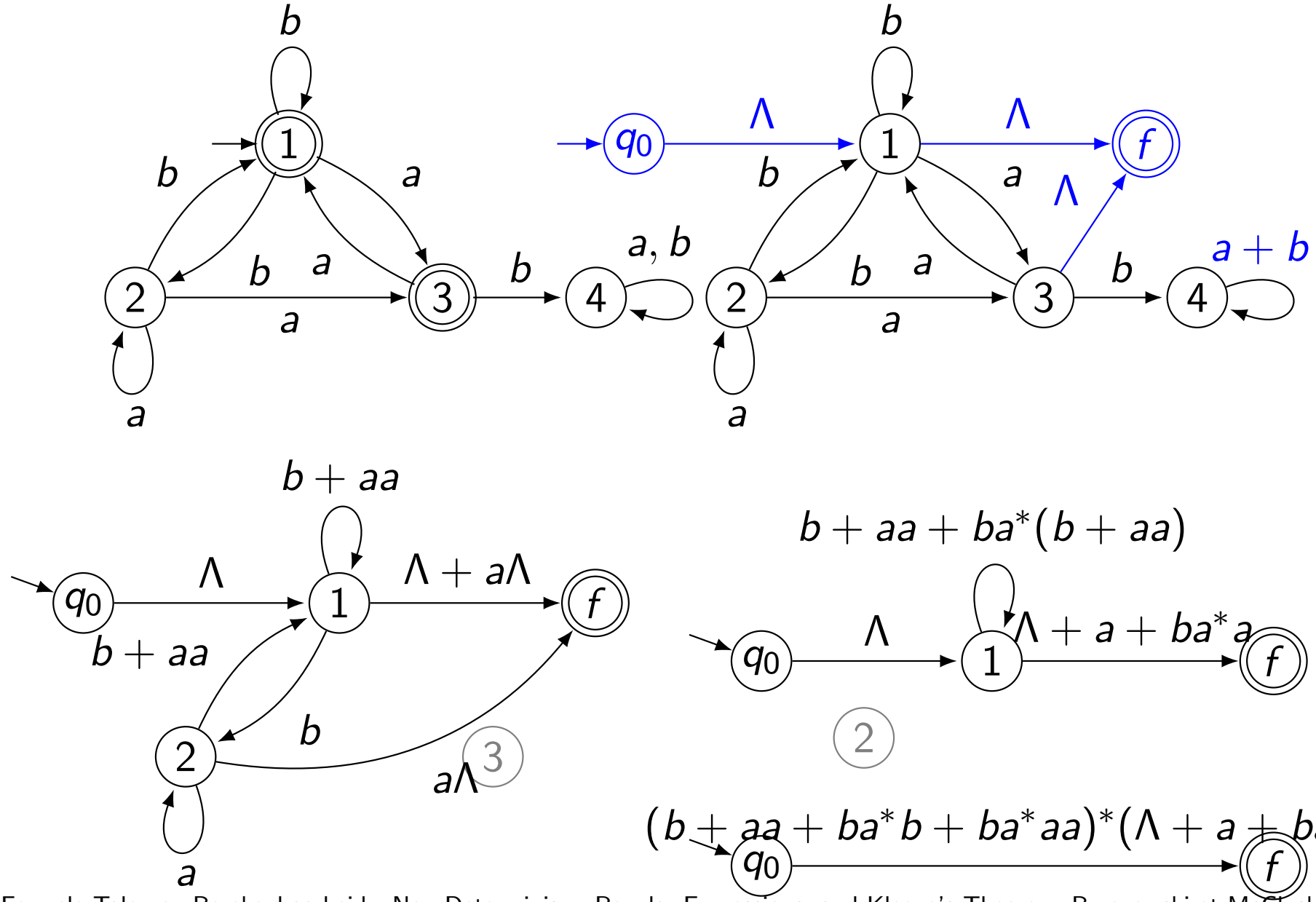
[M] Exercise 3.54

Example



Eliminate 4,3,2,1

Example


b
1
b
a
2
b
a
3
b
a
4
a, b
q0
Λ
1
Λ
f
b
a
Λ
2
b
a
3
b
a + b
4
a
b + aa
q0
Λ
1
Λ + aΛ
f
b + aa
2
b
aΛ
3
a
b + aa + ba*(b + aa)
q0
Λ
1
Λ + a + ba*a
f
2
(b + aa + ba*b + ba*aa)*(Λ + a + ba*a)
q0
f

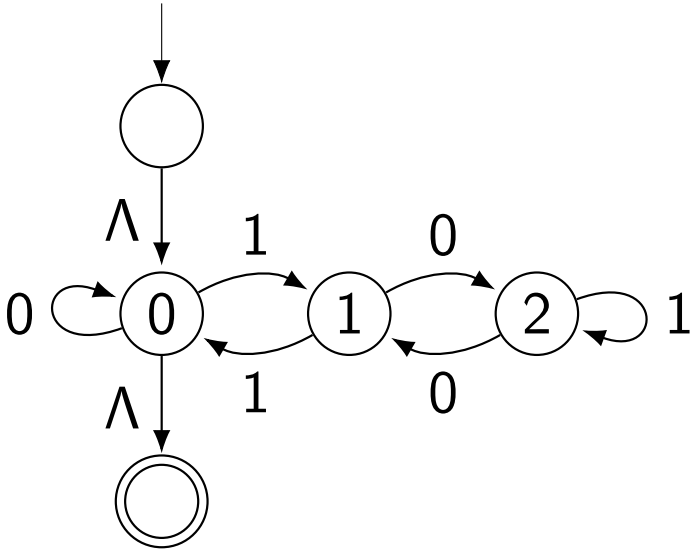
ABOVE

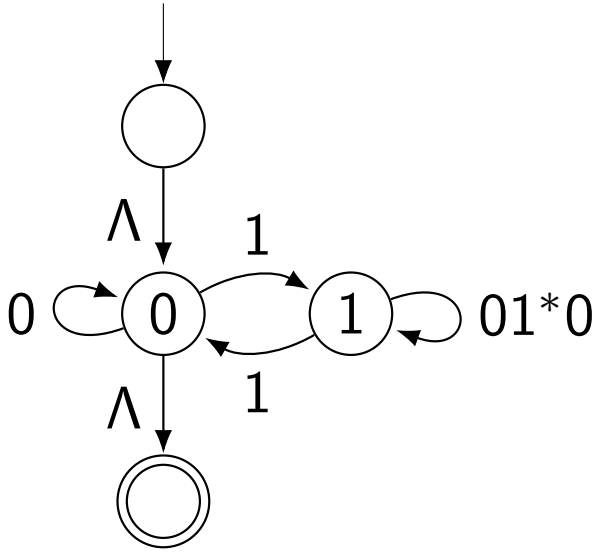
Start by adding new initial and accepting states i and f. Connect these to the original initial and accepting states by edges with the expression Λ. Note we also replaced the parallel edges a, b (loops on node 4) with the expression $a + b$.

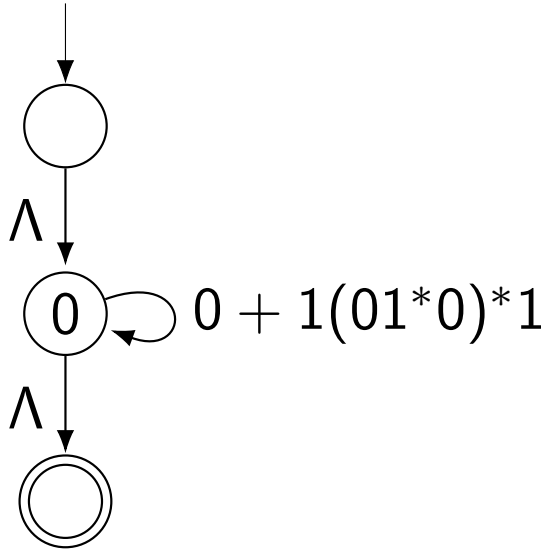
The first node that is eliminated is 4. The proces is not visible here, as there are no pairs (i, j) such that there are edges $(i, R_1, 4)$ and $(4, R_2, j)$, because there are no outgoing edges from 4. Thus no edges are constructed.

The second node eliminated is 3, as shown.

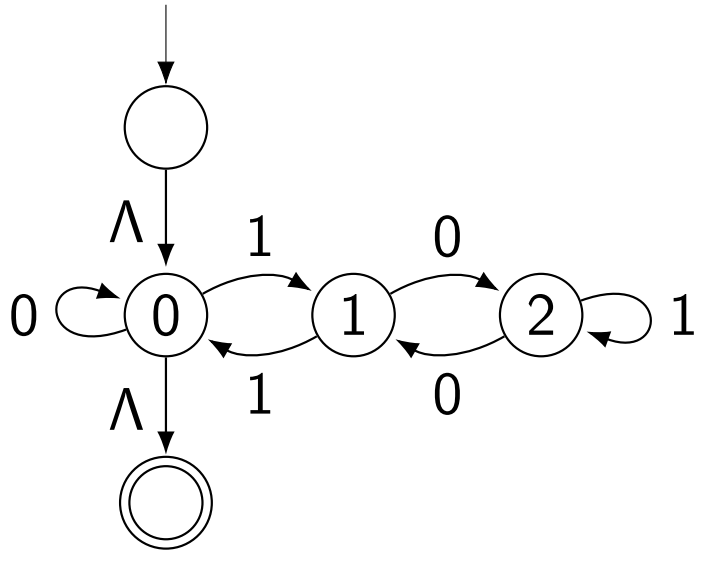
Example divisible by 3, order 1

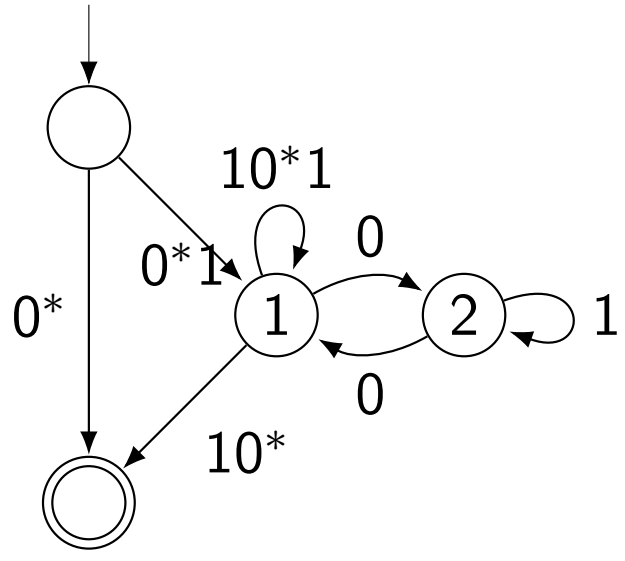


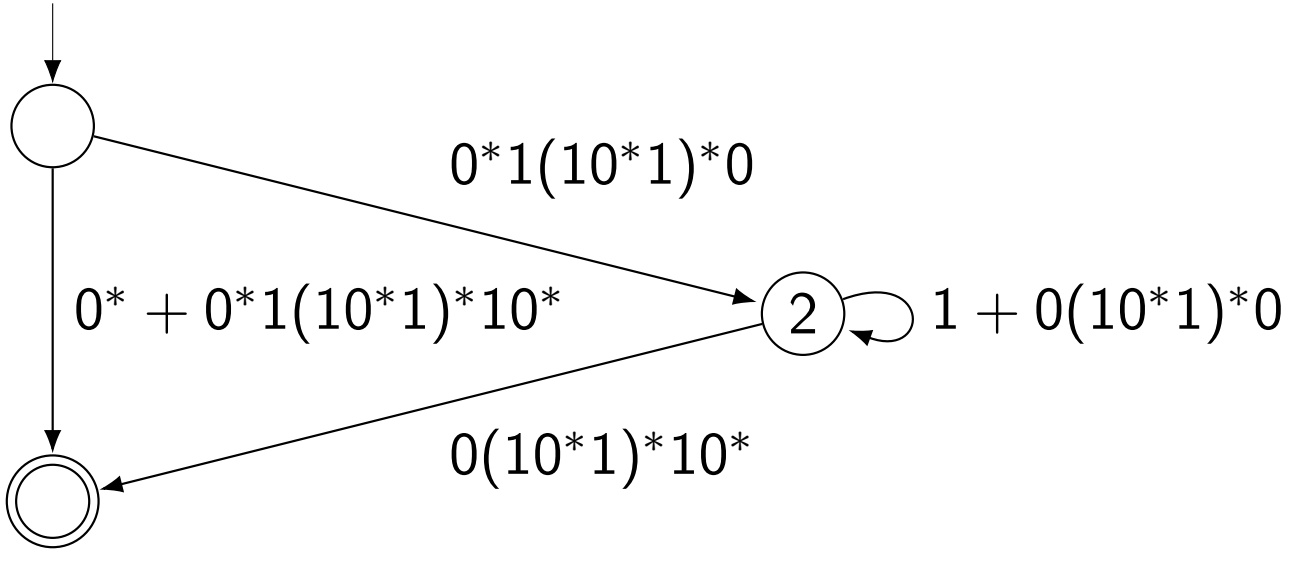




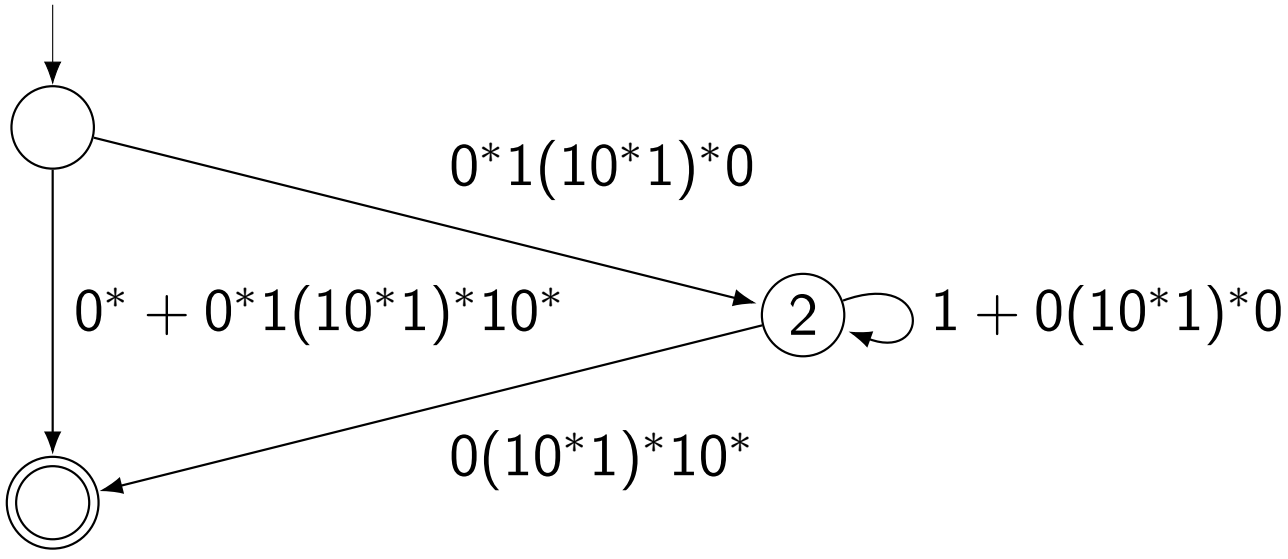
Example divisible by 3, order 2

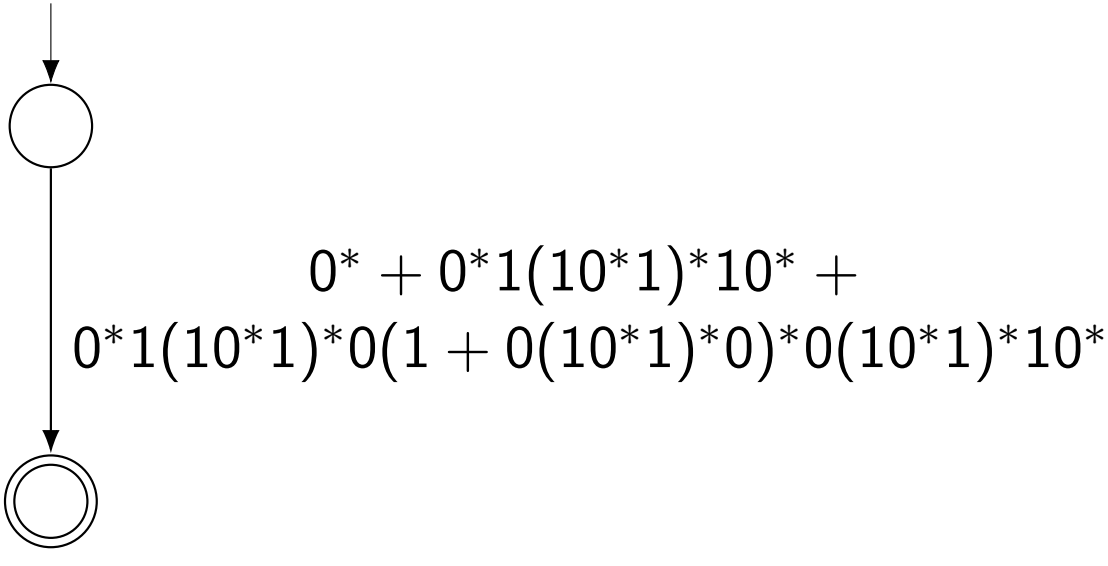






Example divisible by 3, order 2





ABOVE

We compute a regular expression from the given automaton in two different reduction orders.

The first example reduces nodes in the order $2, 1, 0$. The result is
$(0 + 1(01^*0)^*1)^*$
(The removal of the last loop is not depicted.)

The second example in the order $0, 1, 2$. The result is
$0^* + 0^*1(10^*1)^*10^* + 0^*1(10^*1)^*0(1 + 0(10^*1)^*0)^*0(10^*1)^*10^*$
The result differs in structure and size.

Closure

Regular languages are closed under

- Boolean operations (complement, union, intersection, minus)
- Regular operations (union, concatenation, star)
- Reverse (mirror)
- (inverse) Homomorphism (not treated this year)

In the wild

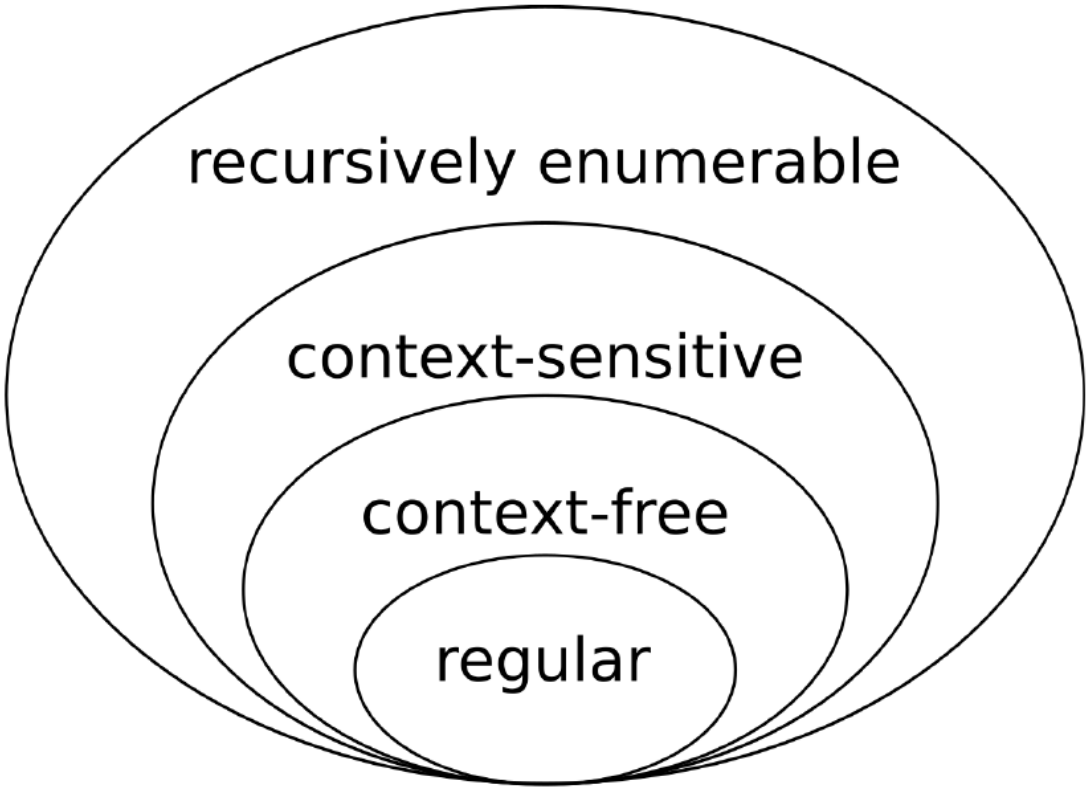
- software engineering, e.g., automata-based modeling language
- modelling of hardware circuits, a book on this
- lexical analysis (compiling high-level language program), e.g., lex
- networks protocols, e.g., TCP/IP, or in packet filtering BGP
- efficient string matching algorithm, e.g., Thompson's algorithm is used in `grep` in Unix
- buttons?
- other thoughts
- ...

Section 4

Context-Free Languages

3 Context-Free Languages
- Examples: recursion
- Regular operations

Chomsky hierarchy



Turing machines

Pushdown automata

Finite automata

[M] Table 8.21

reg. languages	FA	reg. grammar	reg. expression
determ. cf. languages	DPDA		
cf. languages	PDA	cf. grammar	
cs. languages	LBA	cs. grammar	
re. languages	TM	unrestr. grammar	

⟨assignment⟩ ::= ⟨variable⟩ = ⟨expression⟩

⟨statement⟩ ::= ⟨assignment⟩ |
⟨compound-statement⟩ |
⟨if-statement⟩ |
⟨while-statement⟩ | ...

⟨if-statement⟩ ::=
if ⟨test⟩ then ⟨statement⟩ |
if ⟨test⟩ then ⟨statement⟩ else ⟨statement⟩

⟨while-statement⟩ ::=
while ⟨test⟩ do ⟨statement⟩

Propositional logic as a formal language

Definition (well-formed formulas)

. . . by using the construction rules below, and only those, finitely many times:

– every propositional atom $p, q, r, \ldots$ is a wff
– if ϕ is a wff, then so is $(\neg\phi)$
– if ϕ and ψ are wff, then so are $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$,

BNF Backus Naur form

$\psi ::= p \mid (\neg\psi) \mid (\psi \wedge \psi) \mid (\psi \vee \psi) \mid (\psi \rightarrow \psi)$

M.Huet & M.Ryan, Logic in Computer Science

$AnBn = \{ a^n b^n \mid n \geq 0 \} \subseteq \{a, b\}^*$
Recursive definition:

Example

– $\Lambda \in AnBn$
– for every $x \in AnBn$, also $axb \in AnBn$

[M] E 1.18

$AnBn = \{\ a^n b^n \mid n \geq 0\ \} \subseteq \{a, b\}^*$
Recursive definition:

Example

– $\Lambda \in AnBn$ (basis)
– for every $x \in AnBn$, also $axb \in AnBn$ (induction)

$S \to \Lambda$
$S \to aSb$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aa\,bb$$
$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaa\,bbb$$

if $S \Rightarrow^* x$ then also $S \Rightarrow^* axb$
[M] E 4.1

Variants

$AnBn = \{ a^n b^n \mid n \geq 0 \}$

variants

$\{ a^n b^{n+1} \mid n \geq 0 \}$

$S \to b$ (end with extra b)

$S \to aSb$

$\{ a^i b^j \mid i \leq j \}$

$S \to \Lambda$

$S \to aSb \mid Sb$ (free b's)

$\{ a^i b^j \mid i \neq j \}$

$S \to A \mid B$ (choice!)

$A \to aAb \mid aA \mid a$ $(i > j)$

$B \to aBb \mid Bb \mid b$ $(i < j)$

Palindromes

$Pal = \{x \in \{a, b\}^* \mid x^r = x\}$ (reverse)

In canonical (shortlex) order: ...

[M] E 1.18

$Pal = \{x \in \{a, b\}^* \mid x^r = x\}$ (reverse)
In canonical (shortlex) order:
$\Lambda, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, abba, \ldots$
Recursive definition:

Example

– $\Lambda, a, b \in Pal$
– for every $x \in Pal$, also $axa, bxb \in Pal$

[M] E 1.18

$Pal = \{x \in \{a, b\}^* \mid x^r = x\}$ (reverse)
In canonical (shortlex) order:
$\Lambda, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, abba, \ldots$
Recursive definition:

Example

– $\Lambda, a, b \in Pal$
– for every $x \in Pal$, also $axa, bxb \in Pal$

$S \to \Lambda \mid a \mid b$
$S \to aSa$
$S \to bSb$

$S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabSbaa \Rightarrow aababaa$

[M] E 4.3