Finite automata and formal languages (DIT322, TMV028)

Nils Anders Danielsson

2020-02-06

Today

- Regular expressions.
- Translation from finite automata to regular expressions.

Syntax of regular expressions

Syntax

The set $RE(\Sigma)$ of regular expressions over the alphabet Σ can be defined inductively in the following way:

$$\frac{}{\mathsf{empty} \in RE(\Sigma)} \qquad \frac{}{\mathsf{nil} \in RE(\Sigma)}$$

$$\frac{a \in \Sigma}{\mathsf{sym}(a) \in RE(\Sigma)} \qquad \frac{e_1, e_2 \in RE(\Sigma)}{\mathsf{seq}(e_1, e_2) \in RE(\Sigma)}$$

$$\frac{e_1, e_2 \in RE(\Sigma)}{\mathsf{alt}(e_1, e_2) \in RE(\Sigma)} \qquad \frac{e \in RE(\Sigma)}{\mathsf{star}(e) \in RE(\Sigma)}$$

Syntax

Typically we use the following concrete syntax:

$$\frac{}{\emptyset \in RE(\Sigma)} \qquad \frac{}{\varepsilon \in RE(\Sigma)}$$

$$\frac{a \in \Sigma}{a \in RE(\Sigma)} \qquad \frac{e_1, e_2 \in RE(\Sigma)}{e_1 e_2 \in RE(\Sigma)}$$

$$\frac{e_1, e_2 \in RE(\Sigma)}{e_1 + e_2 \in RE(\Sigma)} \qquad \frac{e \in RE(\Sigma)}{e^* \in RE(\Sigma)}$$

(Sometimes $e_1 \mid e_2$ instead of $e_1 + e_2$.)

Syntax

- What if, say, $\varepsilon \in \Sigma$?
- Does ε stand for $\text{sym}(\varepsilon)$ or nil?
- One option: Require that $\emptyset, \varepsilon, +, {}^* \notin \Sigma$.

Syntax

- What does $01 + 2$ mean, $(01) + 2$ or $0(1 + 2)$?
- Sequencing "binds tighter" than alternation, so it means $(01) + 2$.
- Parentheses can be used to get the other meaning: $0(1 + 2)$.
- The Kleene star operator binds tighter than sequencing, so 01^* means $0(1^*)$, not $(01)^*$.

Syntax

- What does $0 + 1 + 2$ mean,
  $0 + (1 + 2)$ or $(0 + 1) + 2$?
- The latter two expressions denote the same language, so the choice is not very important.
- One option (taken by the book):
  Make the operator left associative,
  i.e. choose $(0 + 1) + 2$.
- Similarly 012 means $(01)2$.

Syntax

An abbreviation:

- e^+ means ee^*.
- This operator binds as tightly as the Kleene star operator.

Which of the following statements are correct?

1. $01 + 23$ means $(01) + (23)$.
2. $01 + 23^*$ means $((01) + (23))^*$.
3. $0 + 1^*2 + 3^*$ means $((0 + 1)^*)((2 + 3)^*)$.
4. $0 + 1^*2 + 3^*$ means $(0 + ((1^*)2)) + (3^*)$.
5. 012^*34 means $((((01)(2^*))3)4)$.

Semantics

Semantics

$$
\begin{aligned}
&L \in RE(\Sigma) \to \wp(\Sigma^*) \\
&L(\emptyset) &&= \emptyset \\
&L(\varepsilon) &&= \{\, \varepsilon \,\} \\
&L(a) &&= \{\, a \,\} \\
&L(e_1 e_2) &&= L(e_1)L(e_2) \\
&L(e_1 + e_2) &&= L(e_1) \cup L(e_2) \\
&L(e^*) &&= (L(e))^*
\end{aligned}
$$

Which of the following statements are correct?

1. $abcabc \in L(abc^*)$.
2. $xyyxxy \in L(x(y+x)^*y)$.
3. $\varepsilon \in L(\emptyset^*)$.
4. $110 \in L((\emptyset 1 + 10)^*)$.
5. $\varepsilon \in L((\varepsilon + 10)^+)$.
6. $11100 \in L\Big(\big(1(0+\varepsilon)\big)^*\Big)$.

Regular expression algebra

Regular expression equivalences

- We write $e_1 = e_2$ if $L(e_1) = L(e_2)$.
- Recall that two languages are equal if they contain the same strings.

Which of the following propositions are valid? The alphabet is $\{ 0, 1 \}$.

1. $e + \emptyset = e$.
2. $e\emptyset = e$.
3. $\varepsilon e = e$.
4. $e_1 e_2 = e_2 e_1$.
5. $e_1 + e_2 = e_2 + e_1$.
6. $e + e = e$.
7. $e_1(e_2 + e_3) = e_1 e_2 + e_1 e_3$.
8. $e_1 + e_2 e_3 = (e_1 + e_2)(e_1 + e_3)$.

Regular expression algebra

Regular expressions form a semiring:

$$e + \emptyset = \emptyset + e = e$$
$$e_1 + e_2 = e_2 + e_1$$
$$e_1 + (e_2 + e_3) = (e_1 + e_2) + e_3$$

$$e\varepsilon = \varepsilon e = e$$
$$e_1(e_2e_3) = (e_1e_2)e_3$$

$$e\emptyset = \emptyset e = \emptyset$$
$$e_1(e_2 + e_3) = e_1e_2 + e_1e_3$$
$$(e_1 + e_2)e_3 = e_1e_3 + e_2e_3$$

Regular expression algebra

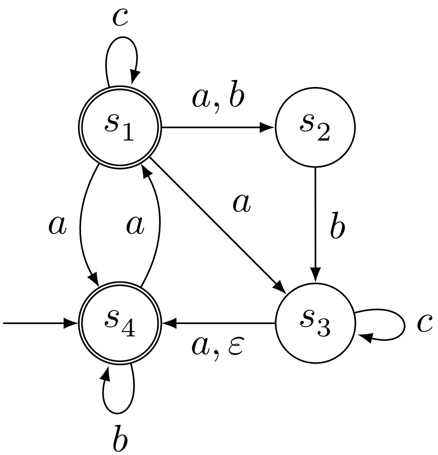
The semiring is idempotent:

$$e + e = e$$

Translating FAs to regular expressions, I

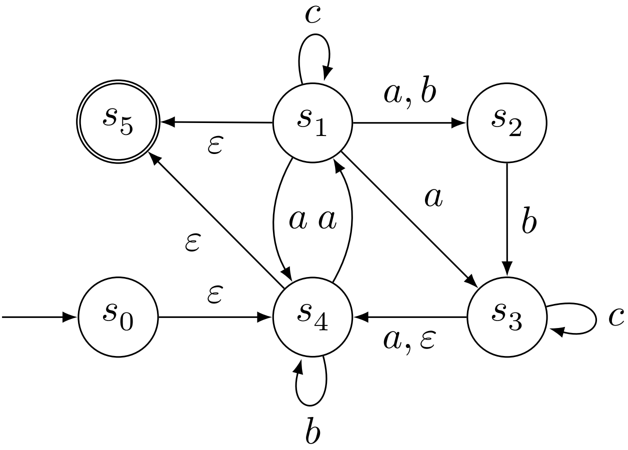
Method one

Consider the following ε-NFA over $\{ a, b, c \}$:



Method one

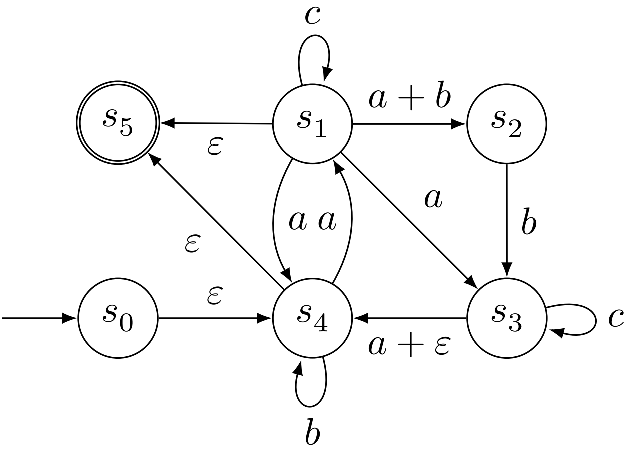
Switch to an equivalent ε-NFA:



(I found this trick in slides due to Klaus Sutner.)

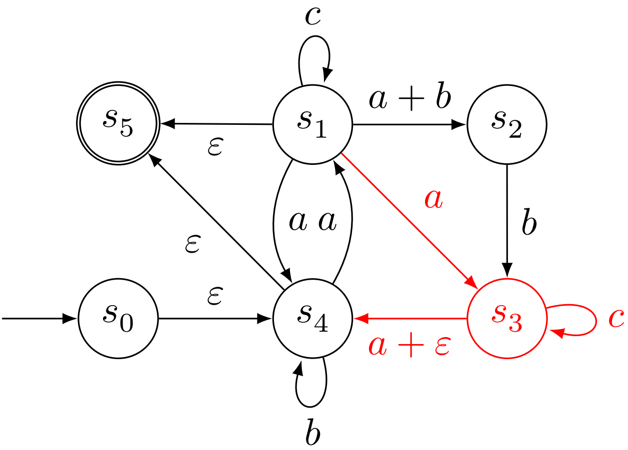
Method one

Turn edge labels into regular expressions:



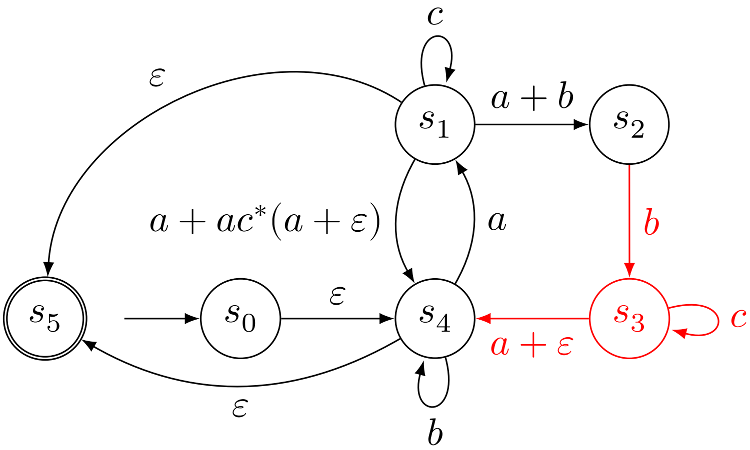
Method one

Eliminate non-accepting states distinct from the start state:



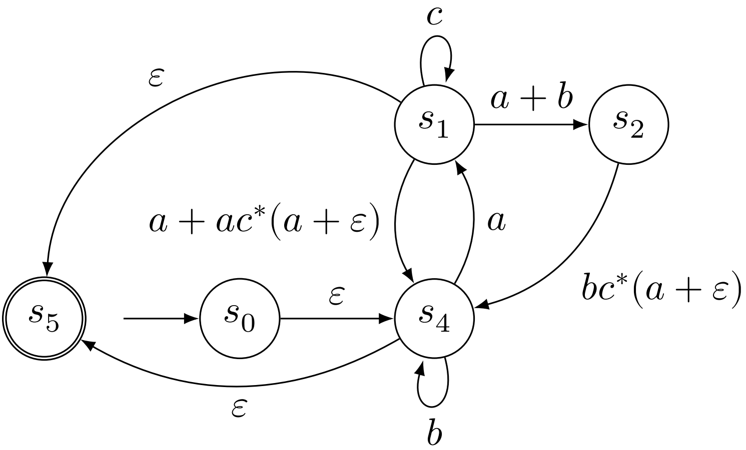
Method one

Eliminate non-accepting states distinct from the start state:



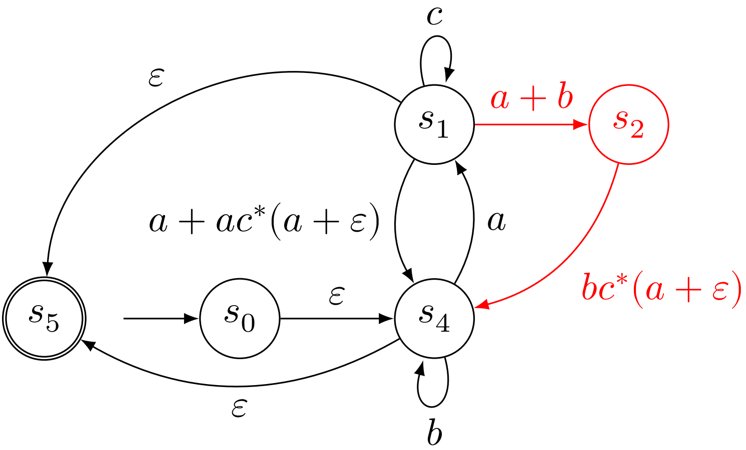
Method one

Eliminate non-accepting states distinct from the start state:



Method one

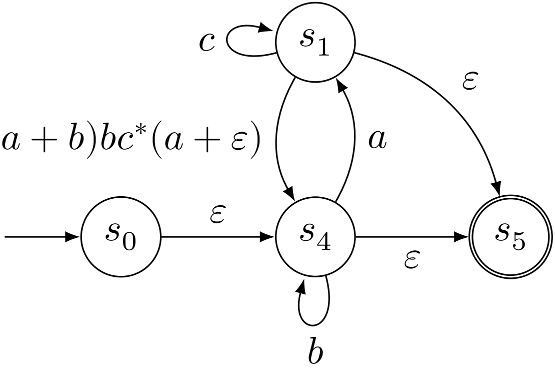
Eliminate non-accepting states distinct from the start state:



Method one

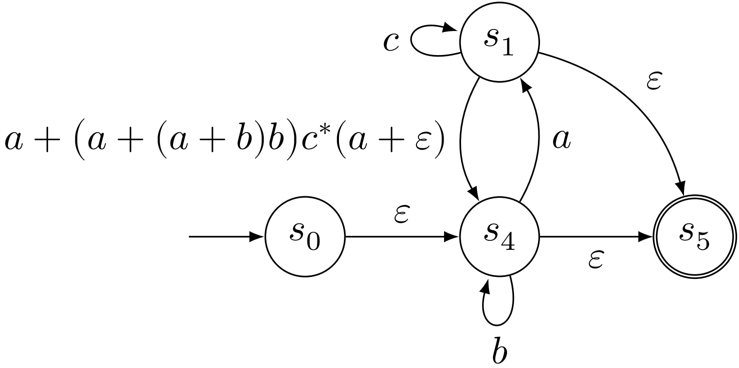
Eliminate non-accepting states distinct from the start state:

$$a + ac^*(a + \varepsilon) + (a + b)bc^*(a + \varepsilon)$$



Method one

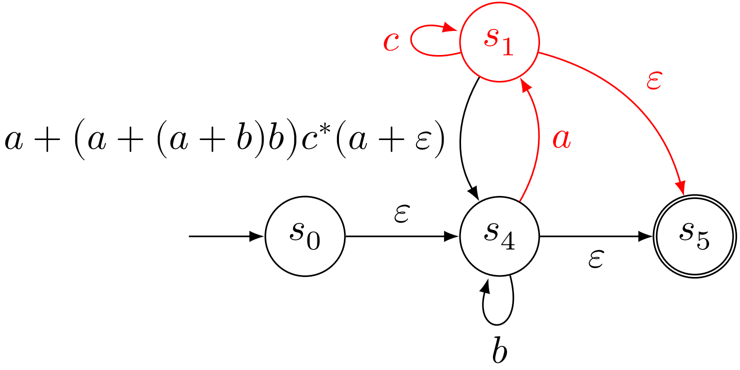
Eliminate non-accepting states distinct from the start state:



It is fine to simplify expressions.

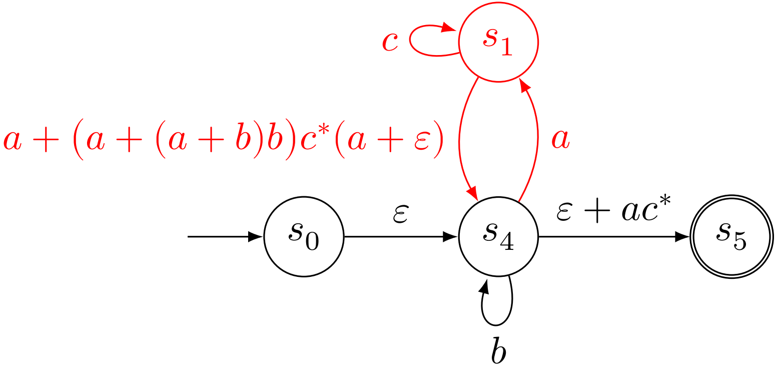
Method one

Eliminate non-accepting states distinct from the start state:



Method one

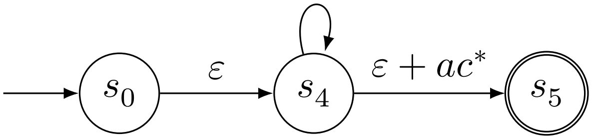
Eliminate non-accepting states distinct from the start state:



Method one

Eliminate non-accepting states distinct from the start state:

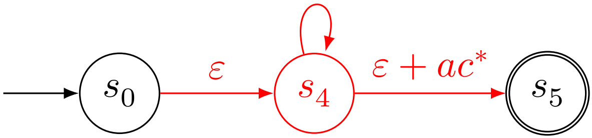
$$b + ac^*\Big(a + (a + (a + b)b)c^*(a + \varepsilon)\Big)$$



Method one

Eliminate non-accepting states distinct from the start state:

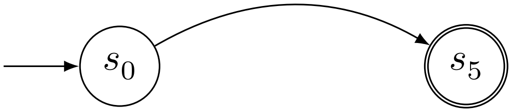
$$b + ac^*\Big(a + (a + (a + b)b)c^*(a + \varepsilon)\Big)$$



Method one

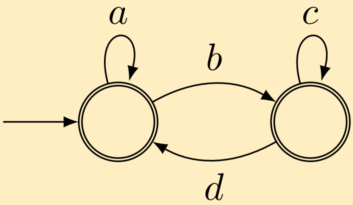
Eliminate non-accepting states distinct from the start state:

$$\Big(b + ac^*\big(a + (a + (a + b)b)c^*(a + \varepsilon)\big)\Big)^*(\varepsilon + ac^*)$$



Done.

Turn the following ε-NFA over $\{ a, b, c, d \}$ into a regular expression.



Translating FAs to regular expressions, II

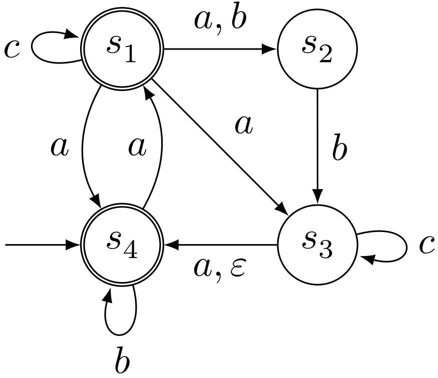
Method two

One form of *Arden's lemma*:

- Let $A, B \subseteq \Sigma^*$ for some alphabet Σ.
- Consider the equation $X = AX \cup B$, where X is restricted to be a subset of Σ^*.
- The equation has the solution $X = A^*B$.
- This solution is the least one (for every other solution Y we have $A^*B \subseteq Y$).
- If $\varepsilon \notin A$, then this solution is unique.

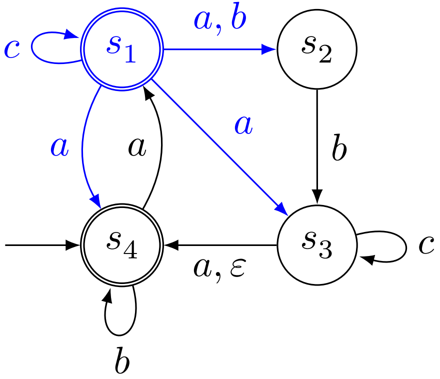
Method two

Consider the following ε-NFA again:



Method two

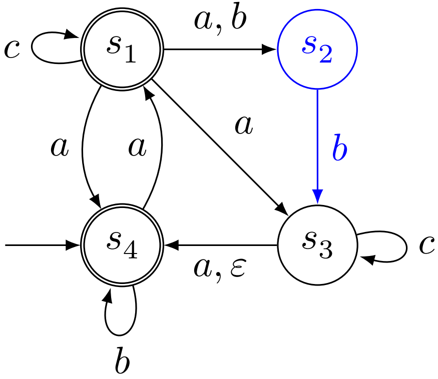
We can turn this ε-NFA into a set of equations.



$$e_1 = \varepsilon + ce_1 + (a + b)e_2 + ae_3 + ae_4$$

Method two

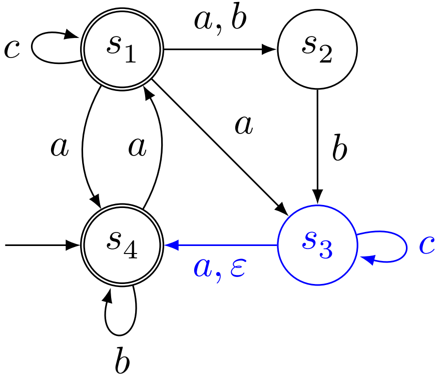
We can turn this ε-NFA into a set of equations.



$$e_2 = be_3$$

Method two

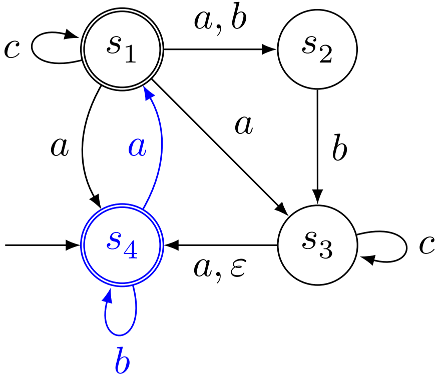
We can turn this ε-NFA into a set of equations.



$$e_3 = ce_3 + (a + \varepsilon)e_4$$

Method two

We can turn this ε-NFA into a set of equations.



$$e_4 = \varepsilon + be_4 + ae_1$$

Method two

Goal: Find the *least* solution for e_4.
(Note that e_4 corresponds to the start state.)

$$
\begin{aligned}
e_1 &= \varepsilon + ce_1 + (a+b)e_2 + ae_3 + ae_4 \\
e_2 &= be_3 \\
e_3 &= ce_3 + (a+\varepsilon)e_4 \\
e_4 &= \varepsilon + be_4 + ae_1
\end{aligned}
$$

Method two

Goal: Find the *least* solution for e_4.
(Note that e_4 corresponds to the start state.)

$$\begin{aligned}
e_1 &= ce_1 + (\varepsilon + (a+b)e_2 + ae_3 + ae_4) \\
e_2 &= be_3 \\
e_3 &= ce_3 + (a+\varepsilon)e_4 \\
e_4 &= be_4 + (\varepsilon + ae_1)
\end{aligned}$$

Eliminate e_2.

Method two

Goal: Find the *least* solution for e_4.
(Note that e_4 corresponds to the start state.)

$$\begin{aligned} e_1 &= ce_1 + (\varepsilon + (a+b)be_3 + ae_3 + ae_4) \\ e_3 &= ce_3 + (a+\varepsilon)e_4 \\ e_4 &= be_4 + (\varepsilon + ae_1) \end{aligned}$$

Method two

Goal: Find the *least* solution for e_4.
(Note that e_4 corresponds to the start state.)

$$\begin{aligned} e_1 &= ce_1 + \Big(\varepsilon + (a + (a+b)b)e_3 + ae_4\Big) \\ e_3 &= ce_3 + (a + \varepsilon)e_4 \\ e_4 &= be_4 + (\varepsilon + ae_1) \end{aligned}$$

Eliminate e_3.

Method two

Goal: Find the *least* solution for e_4.
(Note that e_4 corresponds to the start state.)

$$
\begin{aligned}
e_1 &= ce_1 + \Big(\varepsilon + (a + (a+b)b)e_3 + ae_4\Big) \\
e_3 &= c^*(a+\varepsilon)e_4 \\
e_4 &= be_4 + (\varepsilon + ae_1)
\end{aligned}
$$

Eliminate e_3.

Method two

Goal: Find the *least* solution for e_4.
(Note that e_4 corresponds to the start state.)

$$e_1 = ce_1 + \Big(\varepsilon + (a + (a+b)b)c^*(a+\varepsilon)e_4 + ae_4\Big)$$
$$e_4 = be_4 + (\varepsilon + ae_1)$$

Method two

Goal: Find the *least* solution for e_4.
(Note that e_4 corresponds to the start state.)

$$
\begin{aligned}
e_1 &= ce_1 + \Big(\varepsilon + \big(a + (a + (a+b)b)c^*(a+\varepsilon)\big)e_4\Big) \\
e_4 &= be_4 + (\varepsilon + ae_1)
\end{aligned}
$$

Eliminate e_1.

Method two

Goal: Find the *least* solution for e_4.
(Note that e_4 corresponds to the start state.)

$$\begin{aligned} e_1 &= c^*\Big(\varepsilon + \big(a + (a + (a+b)b)c^*(a+\varepsilon)\big)e_4\Big) \\ e_4 &= be_4 + (\varepsilon + ae_1) \end{aligned}$$

Eliminate e_1.

Method two

Goal: Find the *least* solution for e_4.
(Note that e_4 corresponds to the start state.)

$$e_4 = be_4 + \varepsilon + ac^*\Big(\varepsilon + \big(a + (a + (a+b)b)c^*(a+\varepsilon)\big)e_4\Big)$$

Solve the final equation.

Method two

Goal: Find the *least* solution for e_4.
(Note that e_4 corresponds to the start state.)

$$e_4 = \left(b + ac^*\left(a + (a + (a + b)b)c^*(a + \varepsilon)\right)\right)e_4 + (\varepsilon + ac^*)$$

Solve the final equation.

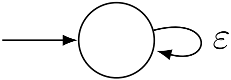
Method two

Goal: Find the *least* solution for e_4.
(Note that e_4 corresponds to the start state.)

$$e_4 = \left(b + ac^*\left(a + (a + (a + b)b)c^*(a + \varepsilon)\right)\right)^*(\varepsilon + ac^*)$$

Method two

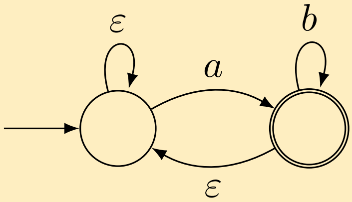
- Why the least solution?
- Consider the following ε-NFA:



- The corresponding equation: $e = \varepsilon e$.
- This equation has infinitely many solutions.
- The least solution gives the right answer:

$$e = \varepsilon^* \emptyset = \emptyset$$

Turn the following ε-NFA over $\{ a, b \}$ into a regular expression.



Today

- Syntax of regular expressions.
- Semantics of regular expressions.
- Regular expression algebra.
- Two methods for translating
  finite automata to regular expressions.

Next week

- Only one lecture.
- Nachi will give the lecture.

Next lecture

- Translation from regular expressions to finite automata.
- More about regular expression algebra.
- The pumping lemma for regular languages.
- Some closure properties for regular languages.