Finite automata theory and formal languages (DIT321, TMV027)

Nils Anders Danielsson

2019-02-07

Today

- Regular expressions.
- Translation from finite automata to regular expressions.
- If there is time: Brzozowski derivatives.

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 $e_1|e_2$ instead of $e_1 + e_2$.)

Syntax

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

Syntax

- What does $01 + 2$ mean, $\mathsf{alt}(\mathsf{seq}(\mathsf{sym}(0), \mathsf{sym}(1)), \mathsf{sym}(2))$ or $\mathsf{seq}(\mathsf{sym}(1), \mathsf{alt}(\mathsf{sym}(1), \mathsf{sym}(2)))$?
- Sequencing "binds tighter" than alternation, so it means $\mathsf{alt}(\mathsf{seq}(\mathsf{sym}(0), \mathsf{sym}(1)), \mathsf{sym}(2))$.
- 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

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

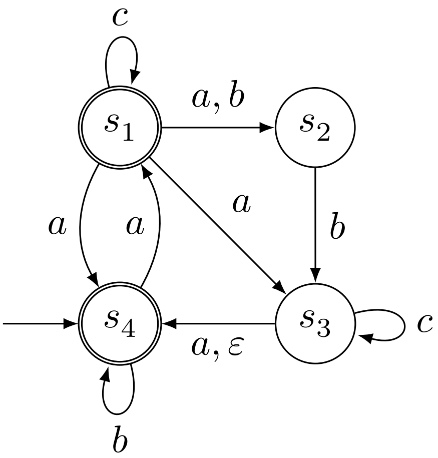
Which of the following statements are correct?

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. $11100 \in L\Big(\big(1(0+\varepsilon)\big)^*\Big)$.

Translating FAs to regular expressions, I

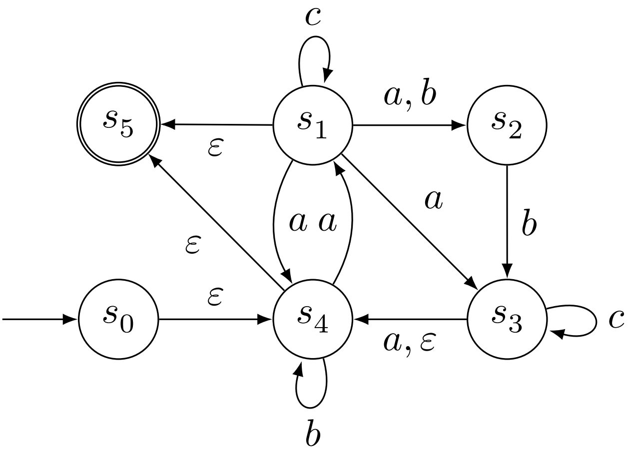
Method one

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



Method one

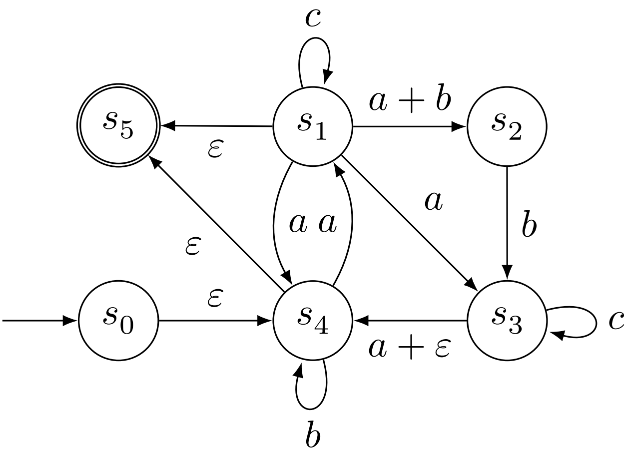
Switch to an equivalent ε-NFA:



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

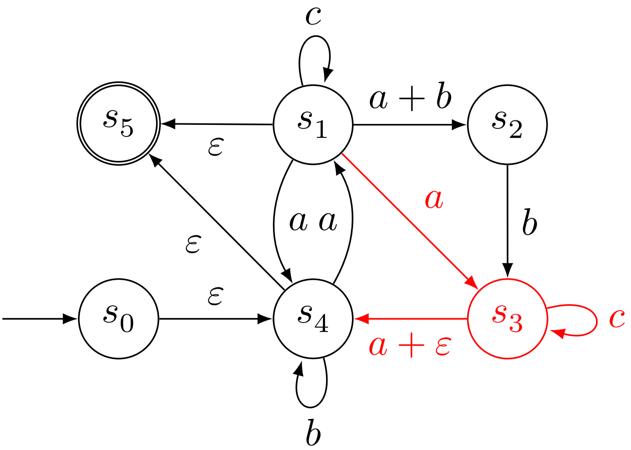
Method one

Turn edge labels into regular expressions:



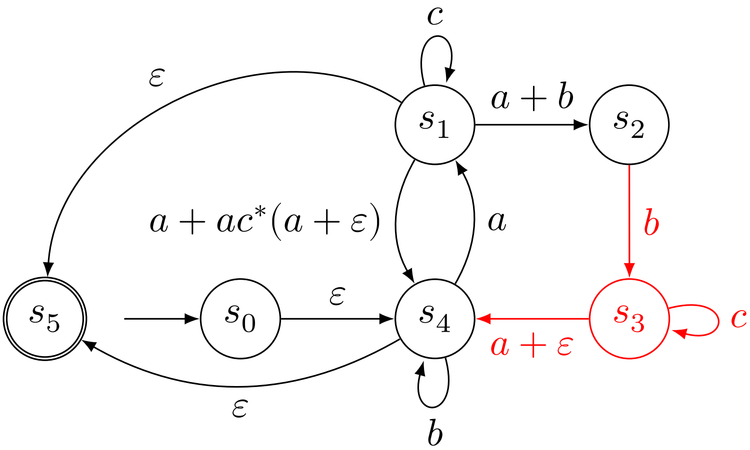
Method one

Eliminate non-accepting states distinct from the start state:



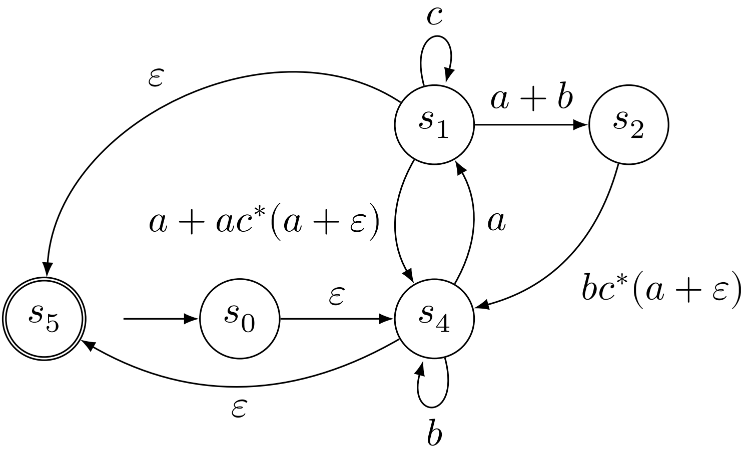
Method one

Eliminate non-accepting states distinct from the start state:



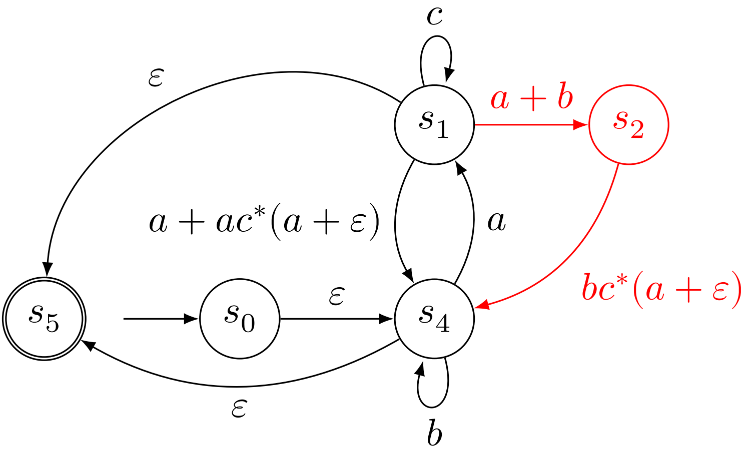
Method one

Eliminate non-accepting states distinct from the start state:



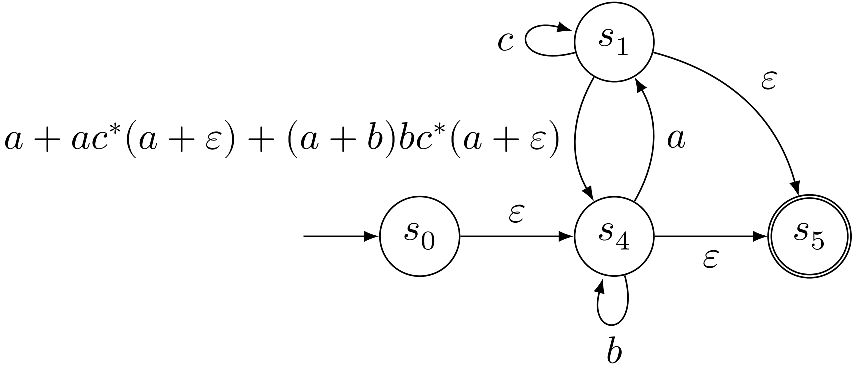
Method one

Eliminate non-accepting states distinct from the start state:



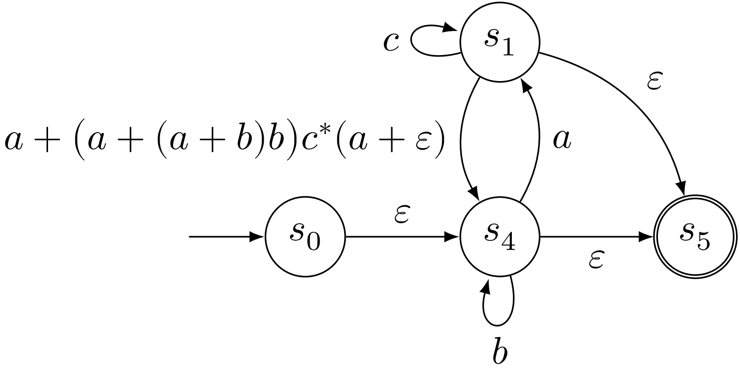
Method one

Eliminate non-accepting states distinct from the start state:



Method one

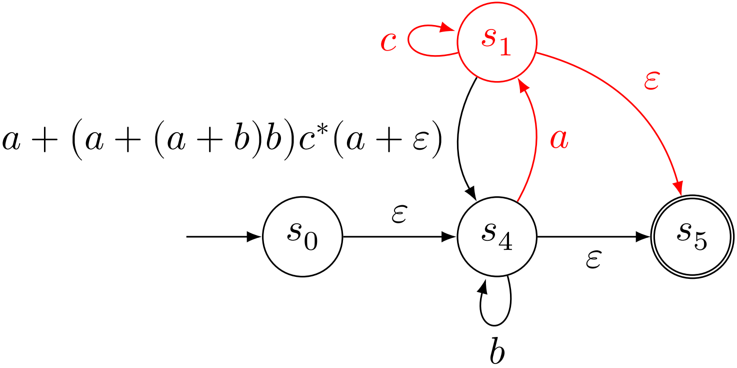
Eliminate non-accepting states distinct from the start state:



It is fine to simplify expressions.

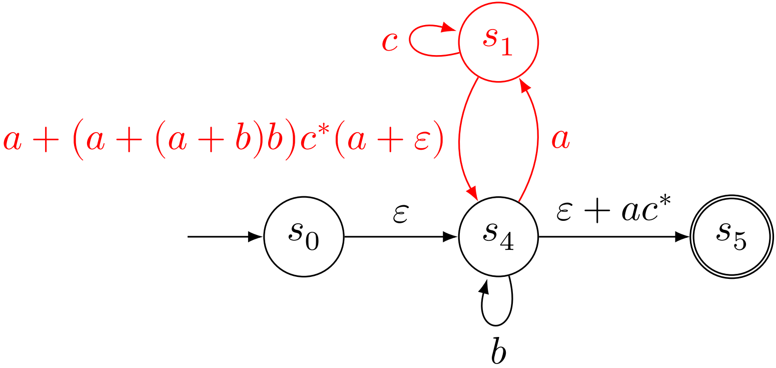
Method one

Eliminate non-accepting states distinct from the start state:



Method one

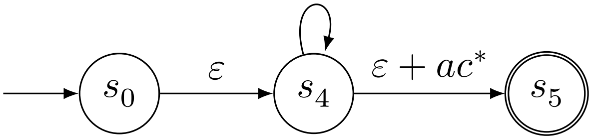
Eliminate non-accepting states distinct from the start state:



Method one

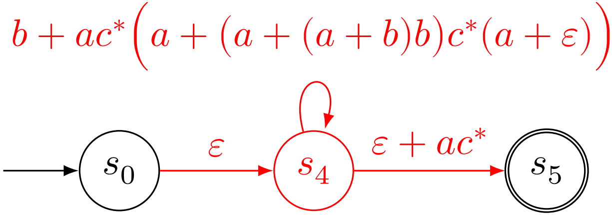
Eliminate non-accepting states distinct from the start state:

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



Method one

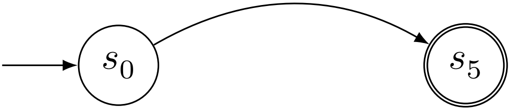
Eliminate non-accepting states distinct from the start state:



Method one

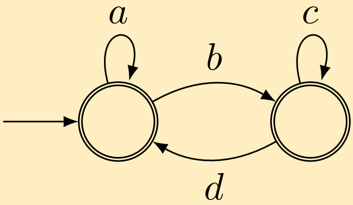
Eliminate non-accepting states distinct from the start state:

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



Done.

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



Translating FAs to regular expressions, II

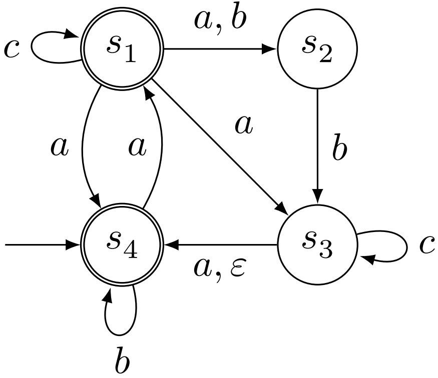
Method two

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.

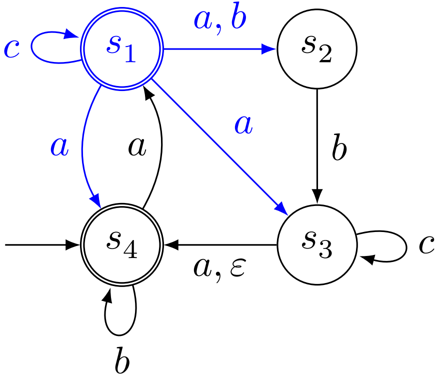
Method two

Consider the following ε-NFA again:



Method two

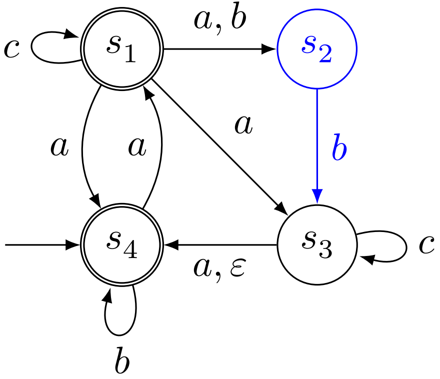
We can turn this ε-NFA into a set of equations.



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

Method two

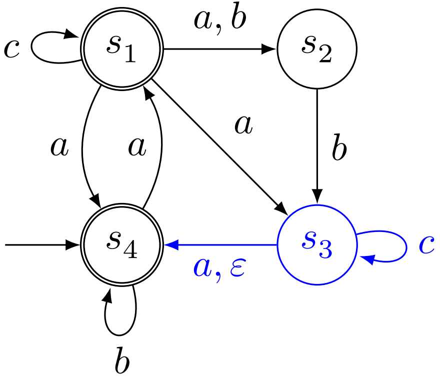
We can turn this ε-NFA into a set of equations.



$$e_2 = be_3$$

Method two

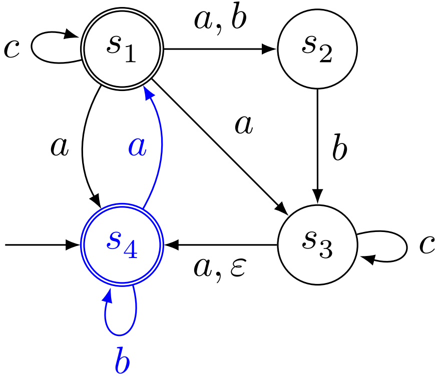
We can turn this ε-NFA into a set of equations.



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

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.)

$$e_1 = ce_1 + \Big(\varepsilon + (a + (a+b)b)e_3 + ae_4\Big)$$
$$e_3 = c^*(a+\varepsilon)e_4$$
$$e_4 = be_4 + (\varepsilon + ae_1)$$

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.)

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

Eliminate e_1.

Method two

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

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

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.)

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

Solve the final equation.

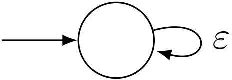
Method two

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

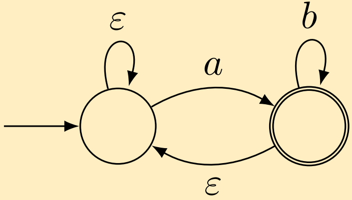
- Why the least solution?
- Consider the following ε-NFA:



- 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.



Brzozowski derivatives

Derivatives

The Brzozowski derivative of a language $L \subseteq \Sigma^*$ with respect to a symbol $a \in \Sigma$:

$$\partial_a(L) = \{ \, w \in \Sigma^* \mid aw \in L \, \}$$

The derivative with respect to a string $w \in \Sigma^*$:

$$\partial_\varepsilon(L) = L$$
$$\partial_{aw}(L) = \partial_w(\partial_a(L))$$

Which of the following languages are equal to $\partial_{01}(\{\,01\,\}^*)$?

1. $\emptyset$.
2. $\{\,01\,\}^*$.
3. $\{\,w \in \{\,0,1\,\}^* \mid 01w \in \{\,01\,\}^*\,\}$.
4. $\{\,w \in \{\,0,1\,\}^* \mid 10w \in \{\,01\,\}^*\,\}$.

Which properties are valid? (All symbols, strings and languages are assumed to be restricted to the same alphabet, $\Sigma = \{ 0, 1 \}$.)

1. $w \in L \Leftrightarrow aw \in \partial_a(L)$.
2. $aw \in L \Leftrightarrow w \in \partial_a(L)$.
3. $w \in L \Leftrightarrow \varepsilon \in \partial_w(L)$.
4. $\varepsilon \in L \Leftrightarrow w \in \partial_w(L)$.
5. $\partial_u(L) = \{ v \in \Sigma^* \mid uv \in L \}$.
6. $\partial_u(L) = \{ v \in \Sigma^* \mid vu \in L \}$.
7. $\partial_a(L^*) = \partial_a(L)L^*$.

Derivatives

- We can check if $w \in L$ by checking if $\varepsilon \in \partial_w(L)$.
- For regular expressions e it is straightforward to compute a regular expression $\partial_w(e)$ satisfying $L(\partial_w(e)) = \partial_w(L(e))$.
- It is also easy to check if a regular expression e is *nullable*, i.e. whether $\varepsilon \in L(e)$.

Derivatives

- Is the regular expression nullable?

$$
\begin{aligned}
&nullable \in RE(\Sigma) \to Bool \\
&nullable(\emptyset) = false \\
&nullable(\varepsilon) = true \\
&nullable(a) = false \\
&nullable(e_1 e_2) = nullable(e_1) \wedge nullable(e_2) \\
&nullable(e_1 + e_2) = nullable(e_1) \vee nullable(e_2) \\
&nullable(e^*) = true
\end{aligned}
$$

- We have $nullable(e) = true$ iff $\varepsilon \in L(e)$.

Derivatives

For $a \in \Sigma$:

$$\partial_a \in RE(\Sigma) \to RE(\Sigma)$$
$$\partial_a(\emptyset) = \emptyset$$
$$\partial_a(\varepsilon) = \emptyset$$
$$\partial_a(a) = \varepsilon$$
$$\partial_a(b) = \emptyset, \text{ if } a \neq b$$
$$\partial_a(e_1 e_2) = \begin{cases} \partial_a(e_1)e_2 + \partial_a(e_2), & \text{if } e_1 \text{ is nullable} \\ \partial_a(e_1)e_2, & \text{otherwise} \end{cases}$$
$$\partial_a(e_1 + e_2) = \partial_a(e_1) + \partial_a(e_2)$$
$$\partial_a(e^*) = \partial_a(e)e^*$$

Derivatives

- Why is the final clause $\partial_a(e^*) = \partial_a(e)e^*$ correct?
- The relevant case of the inductive proof of correctness:

$$L(\partial_a(e^*)) =$$
$$L(\partial_a(e)e^*) =$$
$$L(\partial_a(e))L(e^*) = \{\text{By the inductive hypothesis.}\}$$
$$\partial_a(L(e))L(e^*) =$$
$$\partial_a(L(e))(L(e))^* = \{\text{See a previous quiz.}\}$$
$$\partial_a((L(e))^*) =$$
$$\partial_a(L(e^*))$$

Derivatives

- One can include intersection and complement:

$$\frac{e_1, e_2 \in RE(\Sigma)}{e_1 \cap e_2 \in RE(\Sigma)} \qquad \frac{e \in RE(\Sigma)}{\overline{e} \in RE(\Sigma)}$$

- Exercise: Adapt $nullable$ and ∂.

Today

- Syntax of regular expressions.
- Semantics of regular expressions.
- Two methods for translating
  finite automata to regular expressions.
- Brzozowski derivatives?

Next week

- I will not be here.
- Thomas Sewell will take care of all the scheduled teaching.
- Contact Thomas if you have any urgent questions.

Next lecture

- Translation from regular expressions to finite automata.
- Regular expression equivalences.
- The pumping lemma for regular languages.
- Some closure properties for regular languages.

- Deadline for the next quiz: 2019-02-12, 10:00.
  - Now you get two attempts.
- Deadline for the second assignment: 2019-02-10, 23:59.