Today's lecture: From Patterns to NFAs

Reading:

Today: From Regular Expressions to Finite Automata
Read: *Kozen* lecture 8 or *Sipser* 1.3.

September 23: From Finite Automata to Regular Expressions
Read: *Kozen* lecture 9 or *Sipser* 1.3.

September 26: Nonregular Languages
Read: *Kozen* lecture 12 or *Sipser* 1.4.

September 28: More examples of the pumping Lemma
Read: *Kozen* lecture 13 or *Sipser* 1.4.

September 30: Survey of other topics related to finite automata

October 3: Context Free Languages and Grammars
Read: *Kozen* lecture 19 or *Sipser* 2.1.

October 5: Balanced Parentheses
Read: *Kozen* lecture 20 (or *Sipser* 2.1).

October 7: Chomsky and Greibach normal forms
Read: *Kozen* lecture 21 (or *Sipser* 2.1).

October 10: Thanksgiving, no lecture

October 12: Non-Context-Free Languages
Read: *Kozen* lecture 22 or *Sipser* 2.3.

October 14: Non-deterministic, Pushdown Automata
Read: *Kozen* lecture 23 or *Sipser* 2.2.

October 17: From CFLs to PDAs
Read: *Kozen* lecture 24 (or *Sipser* 2.2).

October 19: From PDAs to CFLs
Read: *Kozen* lecture 25 (or *Sipser* 2.2).

October 21: Deterministic PDAs
Read: *Kozen* lectures E and F.

October 26: Midterm: In class.

I. The big picture

A. What we want to show:

Patterns, regular expressions, NFAs with and without ϵ transitions, and DFAs are all equivalent in terms of the languages that they can recognize.

B. What we have shown:

DFAs and NFAs with and without ϵ transitions are equivalent.

1. Every DFA is an NFA.

2. For every NFA without ϵ transitions, there is a DFA that accepts the same language:

a. Power set construction.
b. Let Q be the set of states of the NFA
c. We construct a DFA whose set of states is 2^Q.
d. Every state of the DFA represents a set of states (possibly empty) of the NFA.
e. This allows the DFA to simulate the NFA.
f. Because $|Q|$ is finite, so is $\left|2^Q\right| = 2^{|Q|}$.

3. Every NFA without ϵ transitions is an NFA where ϵ transitions are allowed (trivial case).

4. For every NFA with ϵ transitions, there is an NFA without ϵ transitions that accepts the same language:

a. If $q \xrightarrow{\epsilon} q'$, then for each arc, $p \xrightarrow{c} q$, add an arc $p \xrightarrow{c} q'$ (if not already present), and delete the arc $q \xrightarrow{\epsilon} q'$.
b. The resulting machine accepts the same language as the original and has one less ϵ transition.
c. Because $|Q|$ is finite, the number of ϵ transitions is finite. Thus, a finite number of applications of this transformation leads to an NFA with no ϵ transitions that accepts the same language as the original NFA. (Note: this is a proof by induction on the number of ϵ transitions in the original NFA).

C. What we need to show:

1. For every pattern, there is a regular expression for the same language.
We'll prove this today.

2. For every regular expression, there is a pattern that recognizes the same language.
This is trivial because regular expressions are a subset of Kozen's patterns. We won't address this one any further.

3. For every pattern, there is an NFA with ϵ transitions that recognizes the same language.
We'll prove this today.

4. For every NFA, there is a regular expression that recognizes the same language. We'll prove this on Friday.

II. From patterns to NFAs

A. For any pattern, α, we construct an NFA that recognizes $L(\alpha)$ based on the structure of α. This provides a proof by induction.

1. Base cases: $a \in \Sigma$, $\boldsymbol{\epsilon}$, $\boldsymbol{\emptyset}$, $\boldsymbol{\epsilon}$:
Each of these can be implemented directly by an NFA as shown in figure 1.

a. $a \in \Sigma$
Let $M_a = (\{q_0, q_1, q_2\}, \Sigma, \delta_a, q_0, \{q_1\})$ where

$$\begin{array}{rcll} \delta_a(q_0, \mathsf{a}) & = & q_1 & \\ \delta_a(q_0, \mathsf{b}) & = & q_2, & \mathsf{b} \in \Sigma - \{a\} \\ \delta_a(q_1, \mathsf{b}) & = & q_2, & \mathsf{b} \in \Sigma \\ \delta_a(q_2, \mathsf{b}) & = & q_2, & \mathsf{b} \in \Sigma \end{array}$$

The proof that $L(M_a) = L(a)$ is straightforward.

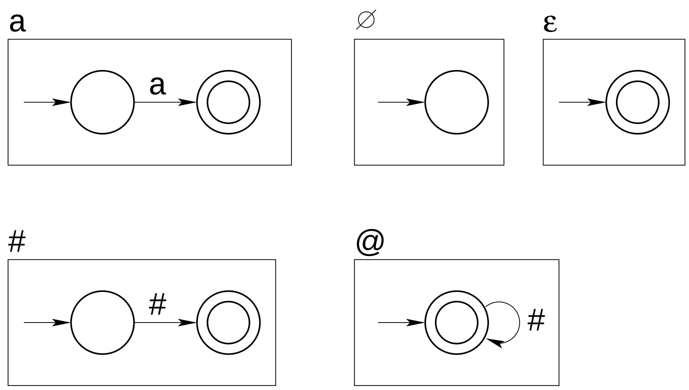

Figure 1: Automata for base cases of proof that every pattern can be realized by an NFA

b. The other cases are similar

2. Induction step:

a. $\beta + \gamma$: see figure 2.

b. $\beta \cap \gamma$: NFAs are closed under intersection. In particular, we can construct DFAs corresponding to the NFAs for β and γ and use the product construction to produce a DFA that recognizes $L(\beta \cap \gamma)$. Because every DFA is an NFA, this completes the construction.

c. $\beta\gamma$: See figure 3.

d. $\sim \beta$: NFAs are closed under complement.

e. β^*: See figure 4.

f. β^+: Note that $\beta^+ = \beta\beta^*$. Now, use the constructions for concatenation and asteration.

3. For every construction of a pattern, we can construct an NFA that recognizes the same language. Thus, every language that can be recognized by a pattern can be recognized by an NFA.

III. Kozen's questions:

A. How hard is it to determine if a given string, x, matches a given pattern, α?

B. Is every language represented by some pattern? Consider

$$B \quad = \quad \{w \in \{\mathsf{a}, \mathsf{b}\}^* | \exists n \in \mathbb{Z}.\ w = a^n b^n\}$$

C. When are two patterns, α and β equivalent, i.e. $L(\alpha) = L(\beta)$?

D. Which operators are redundant?

1. Given $a \in \Sigma$, $+$, $\cdot$, *, and $\boldsymbol{\phi}$,

2. we can construct #, @, $\boldsymbol{\epsilon}$, $\cap$, and $\sim$

a. $\# = c_1 + c_2 + \ldots + c_k$

Where c_1, c_2, $\ldots$, c_k are the elements of Σ. Because Σ is finite, the expression for # is finite as well.

b. $@ = \#^*$

c. $\boldsymbol{\epsilon = \phi^*}$

Note that for any pattern, α, α^0 matches the empty string.

d. $\sim$

As Kozen points out, the proof here is a bit more involved. I'll sketch it in Friday's lecture.

e. $\beta \cap \gamma = \sim (\sim \beta \cup \sim \gamma)$

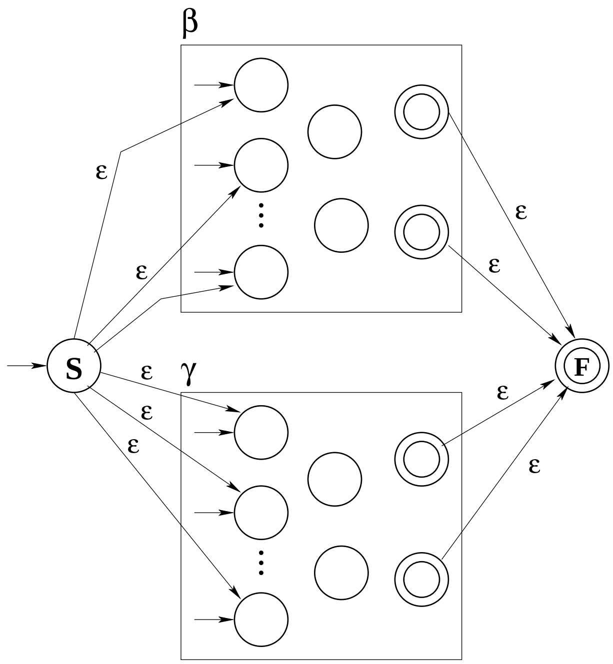

Figure 2: NFA construction for $\beta + \gamma$

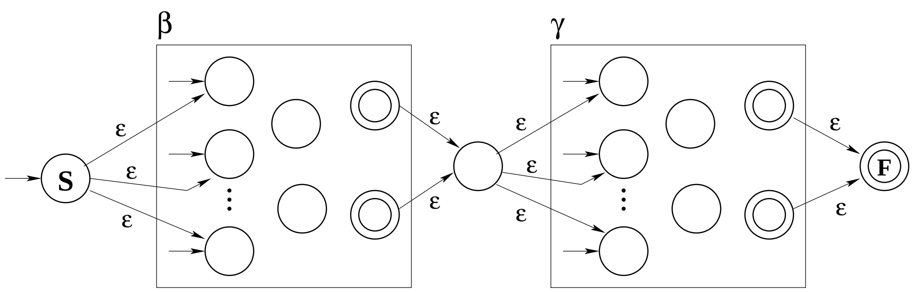

Figure 3: NFA construction for $\beta\gamma$

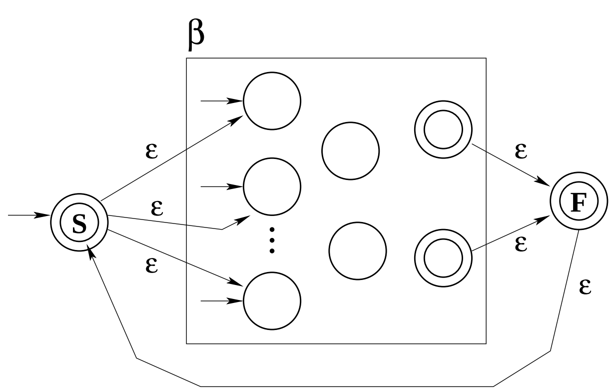

Figure 4: NFA construction for β^*