531 2013-02-07

Note Title 2013-02-07

HW3: Exercise 2.3 [Mogensen]

Lexical Analysis, building on Dr. Fenner's guest lecture, and following Ch. 2 [Mogensen].

Definition 2.1 A Nondeterministic Finite-state Automaton (NFA) consists of:

S - set of states,

$s_0 \in S$ - starting state

$F \subseteq S$ – (set of) accepting states,

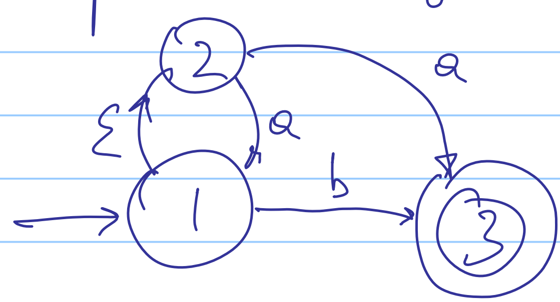
T – set of transitions, which

- connect states
- are labeled with either a symbol from the alphabet (Σ) of the grammar,

  or ε

A transition from state s to state t on symbol c is written $s \xrightarrow{c} t$

Example: (Fig 2.3)



$S = \{1, 2, 3\}$

$s_0 = \{1\}$

$F = \{3\}$

Note the nondeterminism!
- ε
- from 2 on a

$$T = \{1 \xrightarrow{\varepsilon} 2,\ 2 \xrightarrow{a} 1,\ 2 \xrightarrow{a} 3,\ 1 \xrightarrow{b} 3\} \ldots$$

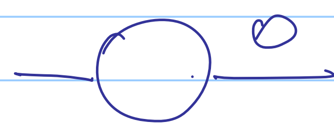
By the way, this NFA recognizes $a^*(a|b)$, whose language is the set of the following strings: b alone, any nonempty sequence of a's, and a

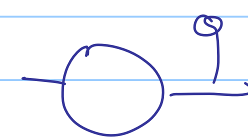
(possibly empty) sequence of a's followed by one b.

A regexp can be converted into an NFA compositionally, i.e., from conversions of the regexp's subexpressions.

The basic rules for the construction are in Fig. 2.4

Example. Build an NFA for $[0-9]^*$

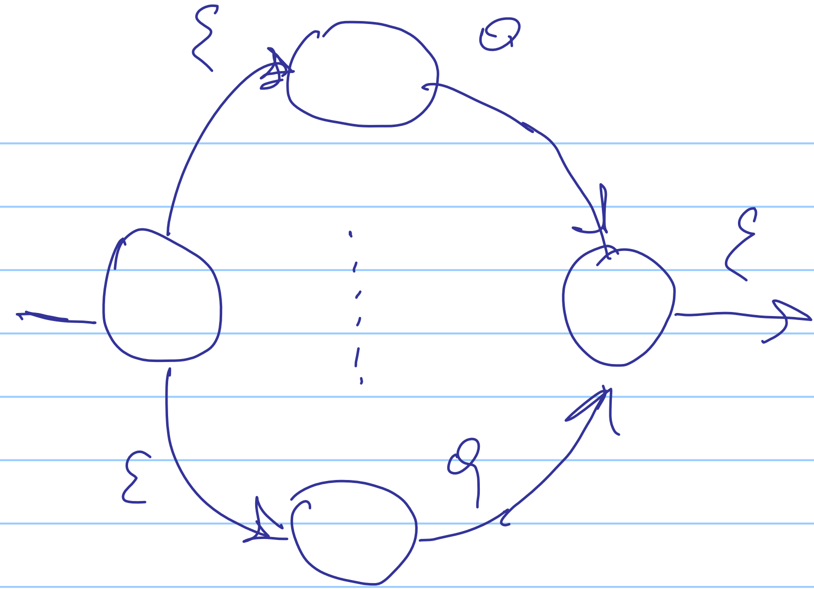
$0 \Rightarrow$ 

, ..., $9 \Rightarrow$ 

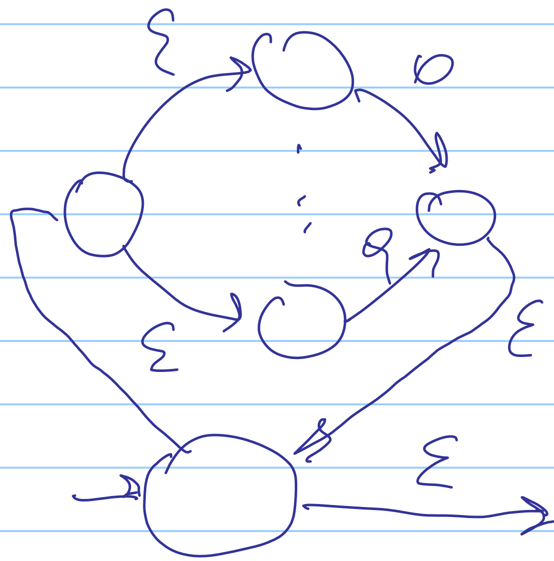
$[0-9]$

$\|$

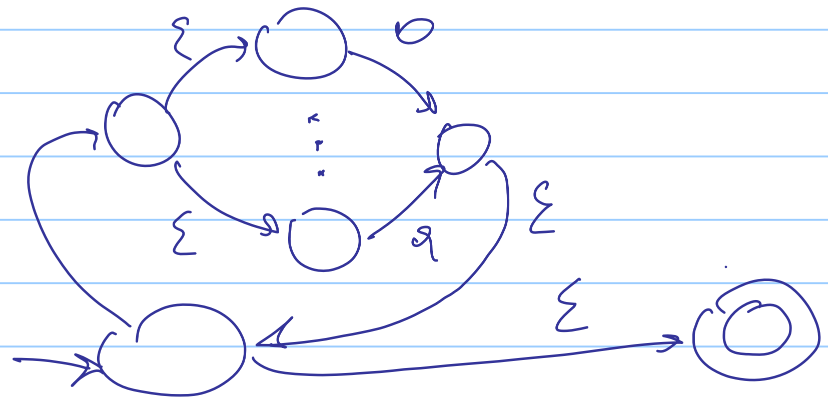
$0 \mid 1 \mid \cdots \mid 9 \Rightarrow$



$[0-9]^* \Rightarrow$

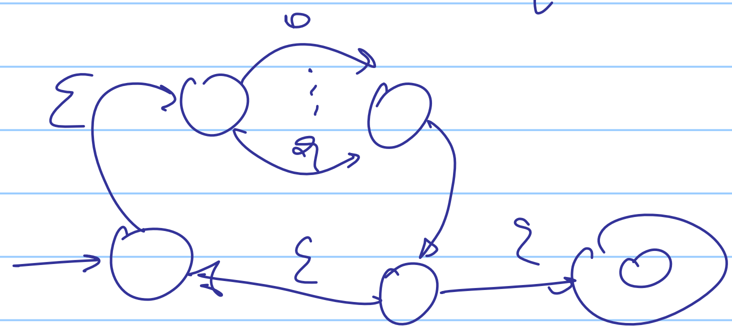


[0-9]* (and no more!)

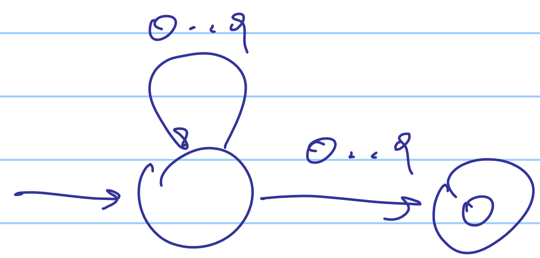


Done.

There are some "optimized" (but not optimal) rules in Figure 2.6. Using the optimized rules to construct $[0-9]^+$, we get:



There is a smaller NFA for $[0-9]^+$:



Deterministic Finite-state Automata (DFA)

are NFAs with two (additional) restrictions:

- there are no ε-transitions
- there may not be two identically labeled transitions out of the same state.

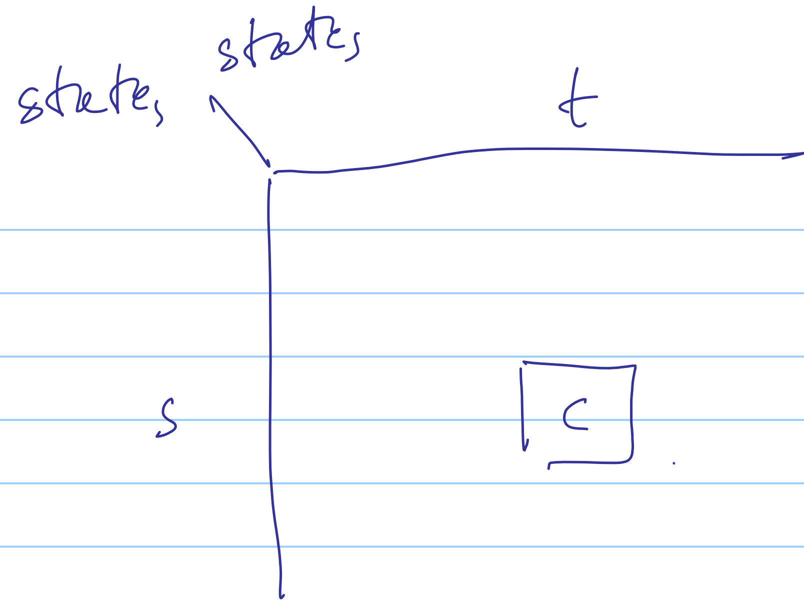
The transition relation $\delta^c t$ in DFAs is

a (partial) function, which we call move:

(partial!)

move(s, c) is the state (if any) reached from s by a transition on symbol c.

A DFA can be implemented by two tables: one represents the accepting states (an array of Booleans). The other represents the move function



The generic entry (s,t) contains symbol c if there is a transition from s to t on symbol c.

Maybe surprisingly, DFA and NFA have the same expressive power — they both recognize regular languages. One way of this equivalence is obvious, b/c every DFA is an NFA. To prove the other way, one should prove the correctness of the algorithm to convert an NFA to a DFA in section 2.6 [Mogensen]. To deal with ε-transitions, we introduce a notation.

Definition 2.2 (ε-closure). Let M be a set of

NFA states. $\underline{\varepsilon\text{-closure}(M)}$ is the least (by set inclusion) solution to the set equation

$$\varepsilon\text{-closure}(M) = M \cup \{ t \mid s \in \varepsilon\text{-closure}(M) \text{ and } s^{\varepsilon}t \in T \},$$ where

T is the set of transitions of the NFA.

Appendix A.4 of Mogensen's book has a detailed discussion of set equations.

A set equation has the form $X = F(X)$.

In our case, we are interested in

ε-closure (M), so

$X = M \cup \{t \mid s \in X \text{ and } s \overset{\varepsilon}{\to} t \in T\}$. So, if we

define F_M to be

$F_M(X) = M \cup \{t \mid s \in X \text{ and } s \overset{\varepsilon}{\to} t \in T\}$, then a solution to $X = F_M(X)$ will be

ε-closure(M).

F is <u>monotonic</u> when, if $X \subseteq Y$, then $F(X) \subseteq F(Y)$.

Solution technique for $X = F(X)$, where F is monotonic.

1. Guess $X = \emptyset$. If $\emptyset = F(\emptyset)$, done

2. Otherwise, try $X = F(\emptyset)$. If true, done

3. Otherwise, try $F(F(\phi))$. Repeat.

An algorithm to convert NFAs to DFAs will be given next time