COP-3402 Systems Software
09/19 Thu

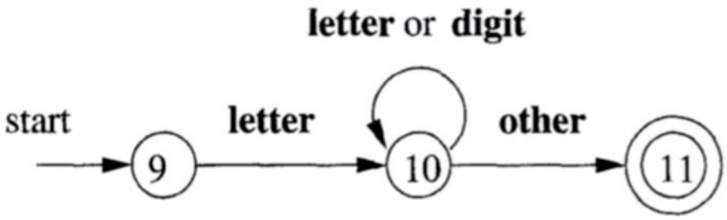
Review:
Regular expressions define string patterns
Concatenation: **ab** **=> a must be followed by b**
Alternation: **a|b** **=> a or b**
Closure: **a*** **=> a; zero or more times**

Given some finite alphabet of symbols, a language is all the possible sequences of symbols, which can be infinite. Regular expressions are a way to define a language.

Regular languages can be represented one for one with finite automata.

Finite automata: A finite set of symbols, and FA can process them one at a time and changes its state depending on what symbols it sees, also how causes it to transition to a new state.
Computational model of turnstile:
Two states: locked and unlocked; changing its state depending on what input/symbol you give.

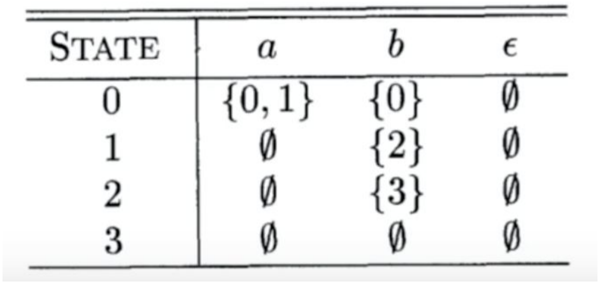
Example of a State Diagram:



Circles represent the **states**, they can be labeled, such as 9, etc. **Transitions** between states are represented by **arrows** between states. Seeing a **letter** transitions us to the next state in our language. Accepting state is represented by a **double circle**.

Epsilon represents empty string in our language.
Deterministic FA: one state at a time
Non-deterministic FA: multiple states at once, the same symbol, multiple transitions, epsilon transitions

Example of FA with Transition Table:

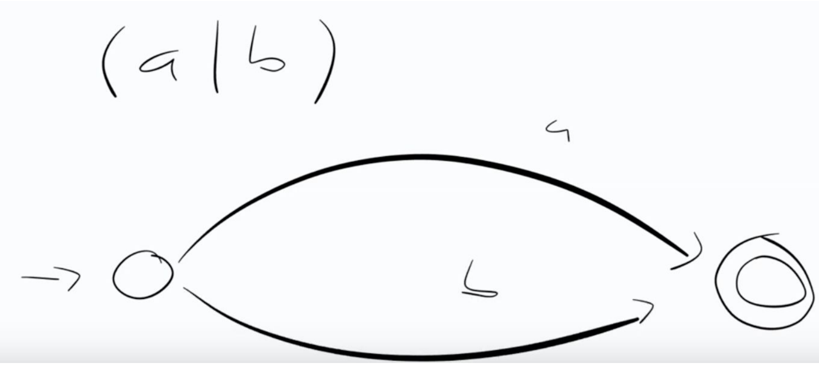


STATE	a	b	ϵ
0	$\{0,1\}$	$\{0\}$	$\emptyset$
1	$\emptyset$	$\{2\}$	$\emptyset$
2	$\emptyset$	$\{3\}$	$\emptyset$
3	$\emptyset$	$\emptyset$	$\emptyset$

=> this ex. Non-deterministic FA.

A non-deterministic FA can be converted to DFA via **subset construction**.
If we have some number n of states, how many subsets of those states are there? 2^n
We can turn all those subsets into a single states in a DFA (subset construction).

Regex Operations as NFAs:



Ex: a|b

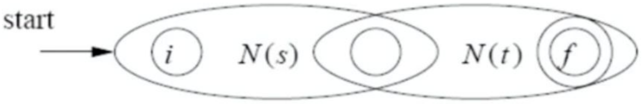


Figure 3.41: NFA for the concatenation of two regular expressions

Dragon book-concatenation.

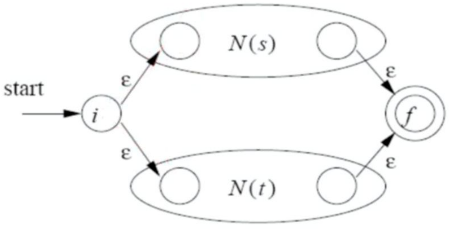


Figure 3.40: NFA for the union of two regular expressions

Dragon book-alternation.

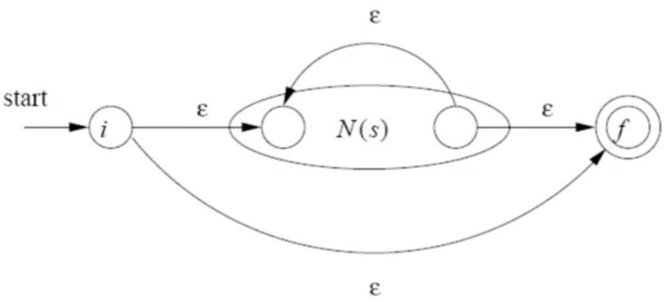
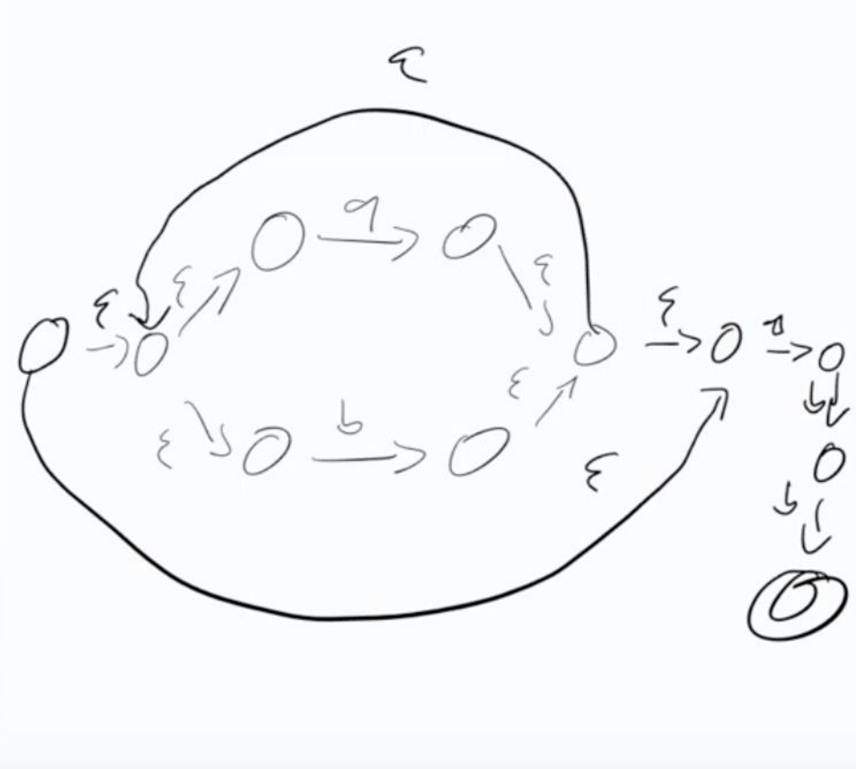


Figure 3.42: NFA for the closure of a regular expression Dragon book-kleene closure.

Ex. (a|b)*abb



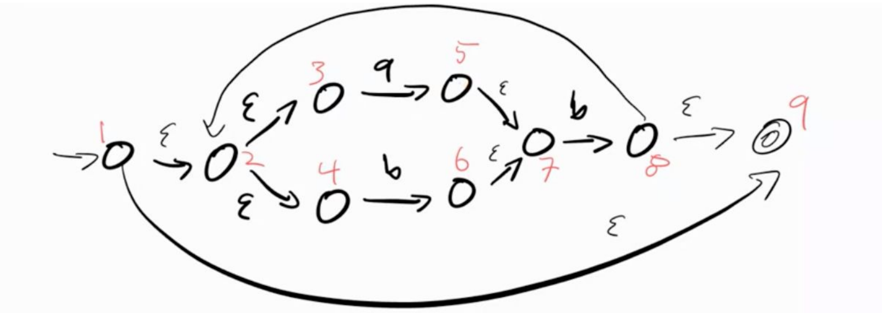
Dragon book.

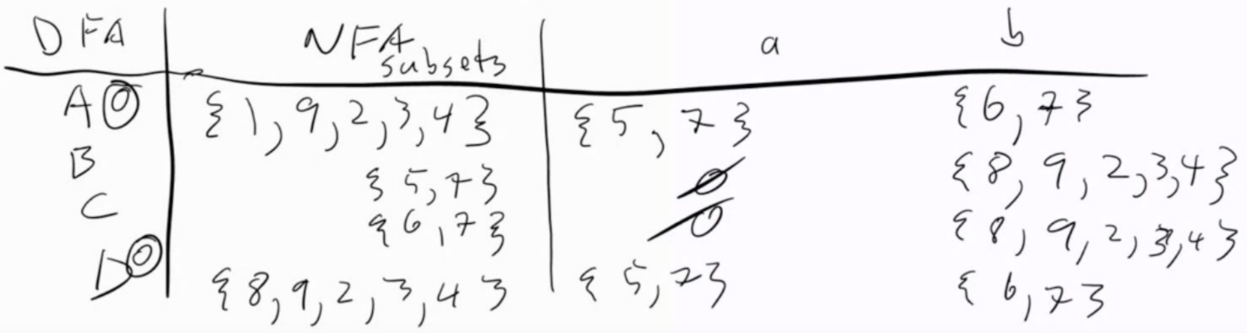
How to construct a DFA from an NFA systematically:
Each DFA state created by subset of NFA states, can be in multiple states
Simulate being in multiple states using a single state
Multiple states are a subset of the NFA states
Create the DFA by calling each subset a single DFA state

Subset Construction Algorithm:

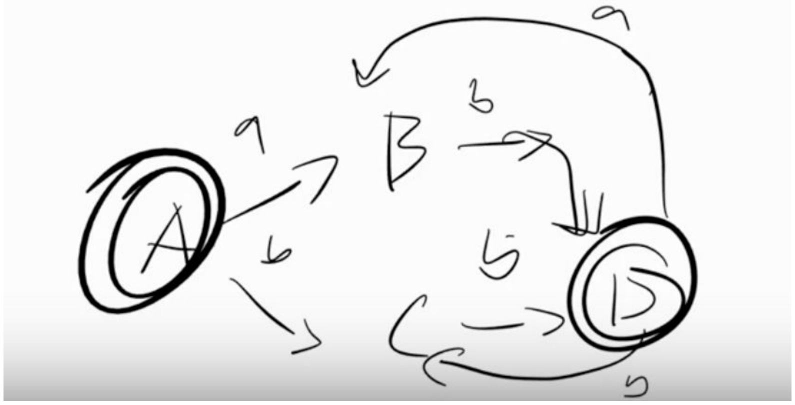
```
Start at the starting state of NFA
Group all states reachable by epsilon (epsilon closure);  call this group as initial state of DFA
For each symbol s in the alphabet;
        Get all states that s transitions to
        Find epsilon-closure of those states
        Call this group as a single state of DFA
Repeat for all combinations of NFA states and symbols
Stop when you have covered them all.
```

Ex of converting from NFA to DFA.
((a|b)b)*





DFA	NFA subsets	a	b
A◎	{1,9,2,3,4}	{5,7}	{6,7}
B	{5,7}	∅	{8,9,2,3,4}
C	{6,7}	∅	{8,9,2,3,4}
D◎	{8,9,2,3,4}	{5,7}	{6,7}


A
B
C
D
a
a
b
b
b
b