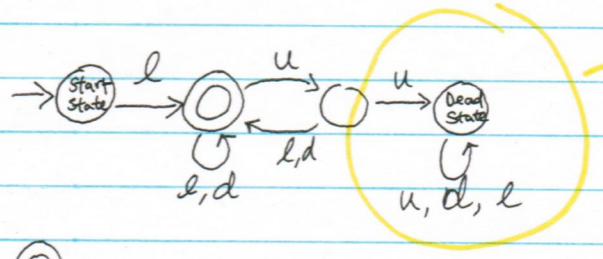
State transition diagrams (Ch 8)

Ex Specification of identities
- first character is a letter ℓ
- remaining characters: letters, digit, underscores d u
- an underscore cannot be the last character and cannot be followed by another underscore

$\Sigma = \{\ell, d, u\}$



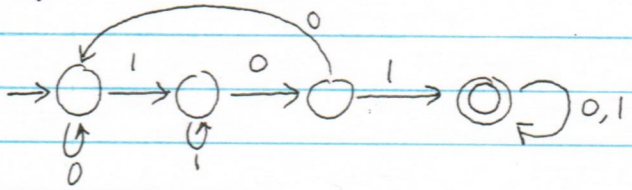
this can be omitted usually.

◎ is an accepting state

Ex $\Sigma = \{0, 1\}$

All strings that have substring 101

Regular expression: $(0+1)^* 101 (0+1)^*$



Definitions:

- Deterministic state transition diagram is a 5-tuple $(\Sigma, Q, q_0, F, \delta)$

where:

Σ is an alphabet

Q is a finite set of states

$q_0 \in Q$ is the start state

$F \subseteq Q$ is the set of accepting state

Hilroy

$\delta : Q \times \Sigma \to Q$ is the transition function

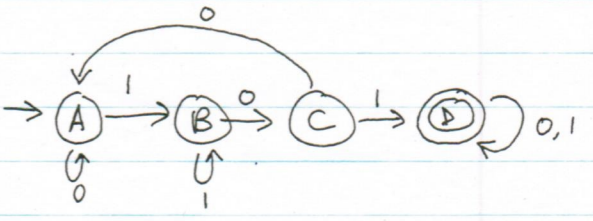
Also known as DFA (deterministic finite automaton)

The transition function can be depicted as a transition table.

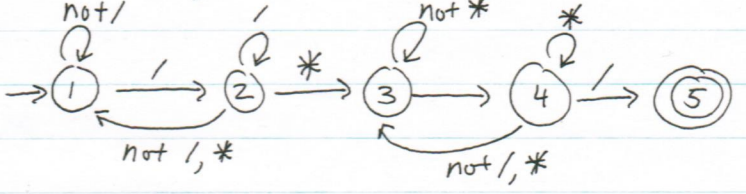
ex

	0	1
A	A	B
B	C	B
C	A	D
D	D	D

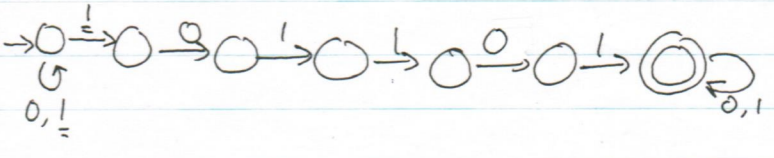
is the table for



Ex DFA that recognizes multiline comments /* ... */



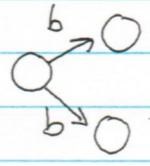
Ex All strings that have substring 10110



Transition state is allowed to be multivalued
Accepts all strings that label some path from the start state to an accepting state.

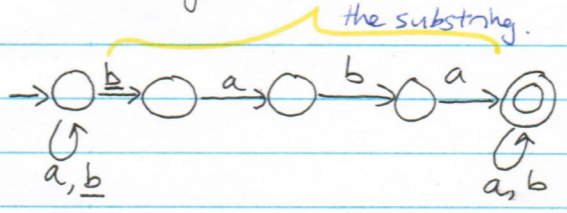
Note: When we want to implement something we need a DFA.
There is an algorithm that converts an NFA to DFA.

Nondeterministic state transition diagram (NFA)



EX $\Sigma = \{a, b\}$

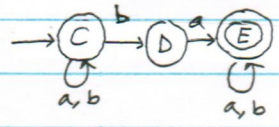
All substrings that have the substring baba



Algorithm that converts an NFA to a DFA : subset construction

The DFA keeps track of the set of possible states that the NFA may be in.

ex NFA:



$\Sigma = \{a, b\}$

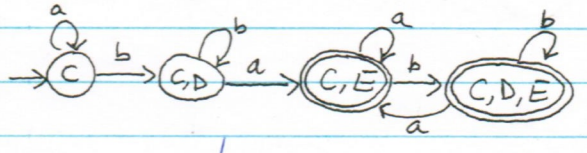
All strings with substring ba

Subset construction:

	a	b
C	{C}	{C, D}
C, D	{C, E}	{C, D}
C, E	{C, E}	{C, D, E}
C, D, E	{C, E}	{C, D, E}

- Begin with start state.
- Produce new states only when new states are reached.
  - ↳ no {E, D}, {E}, {D}

DFA



All states with an E are accepting states (accepting state in NFA)

As you can see, the construction of a DFA from NFA results in a larger state transition diagram.

↳ Thankfully not all states produced are needed.

The above DFA is not the simplest form. We could merge the last 2 states since once substring is made, it can

Hilroy

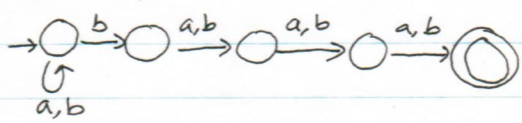
be followed by anything.

EX Sometimes the resulting DFA can be very large.

$\Sigma = \{a, b\}$

All strings where the 4th symbol from the end is "b"

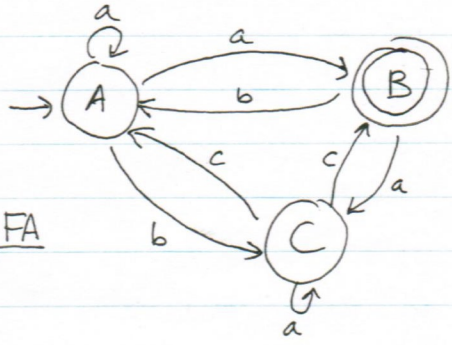
NFA:



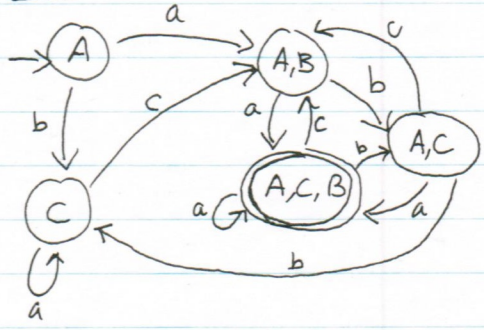
DFA:

Has 15 states!

EX



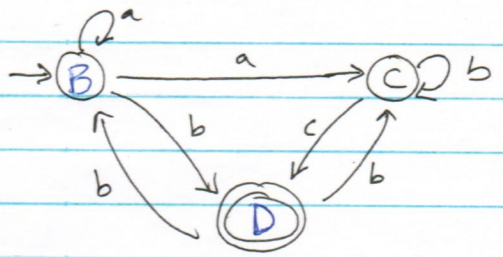
DFA



* Note: Subset construction → the order of sets doesn't matter

Q4

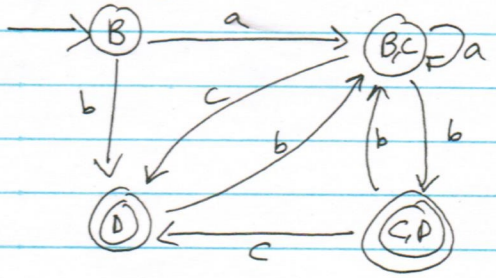
NFA:



Someone brought this up as a question so he explained it @ start of class.

DFA:

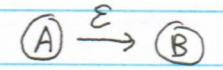
In a complete DFA → all transitions are defined.
→ omitting dead states simplifies. (incomplete DFA)



All undefined transitions go to the dead state $\emptyset$
↳ statement for an incomplete DFA that allows us to exclude dead states

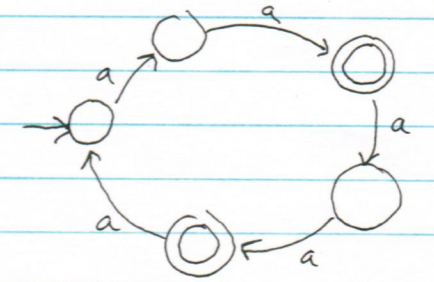
---

NFAs with ε-transitions



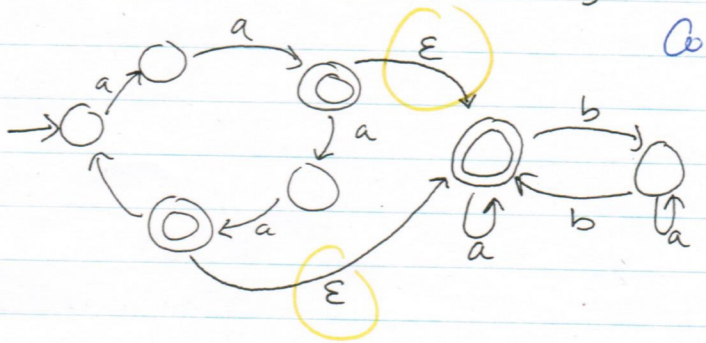
Ex:

$\Sigma = \{a, b\}$ $R = \{a^i \mid i \equiv 2 \pmod 5 \text{ or } i \equiv 4 \pmod 5\}$



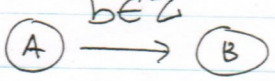
Hilroy

$S = \{ w \in \Sigma^* \mid w \text{ has an even \# of } b\text{'s} \}$

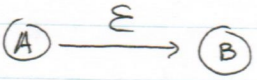


Concatenation $R \cdot S$

ε-transitions explained:



must read b to make transition from A to B



we can choose whether or not to move from A to B. (nondeterministic)

Note:

DFAs cannot have ε-transitions

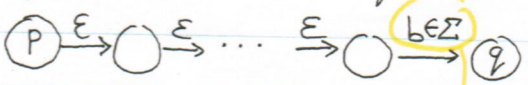
↳ because adding ε-transitions makes it nondeterministic in nature!

Algorithm that for a given ε-NFA M constructs an equivalent NFA without ε-transitions

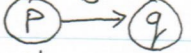
↳ means that we can construct real state diagram that can be implemented.

1. Make a copy M' of M, where all ε-transitions are removed. Also remove all states where incoming are labeled ε, except do not remove start state.

2. Always when M has a sequence of transitions:



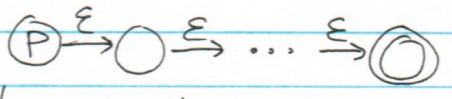
real transition

We should add to M' a transition 

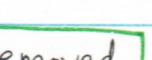
Here p, q may be any states (including the same state)

Hilroy

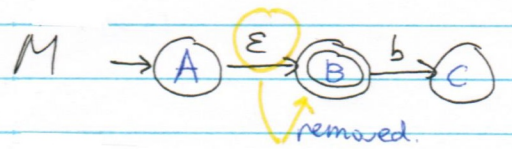
3. Always when we have a situation in M

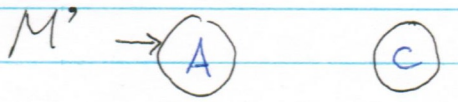


then in M', we should make P an accept state.

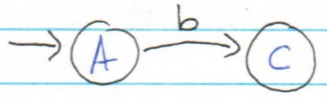
Not removed


Ex. 1

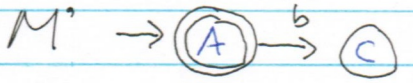




Stage 1: Remove all ε-transitions.

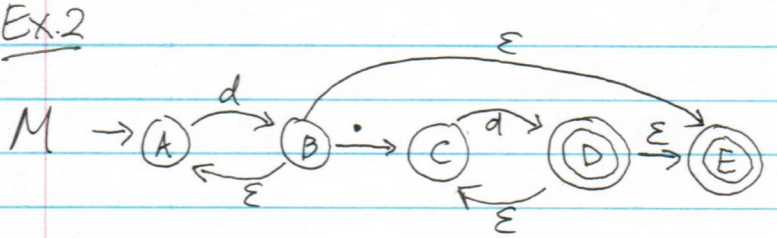


Stage 2: add transitions

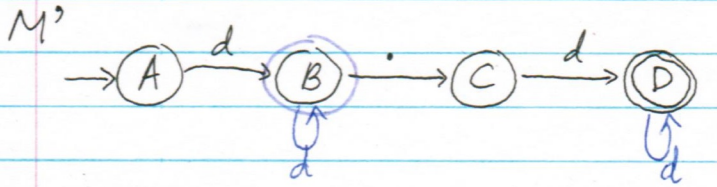


Stage 3: accept states
Now we're done!

Ex. 2

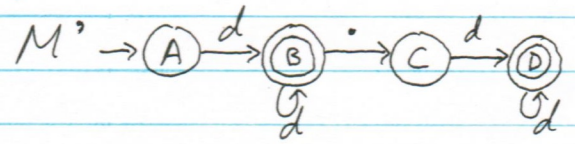


(d is digits
• decimal point)

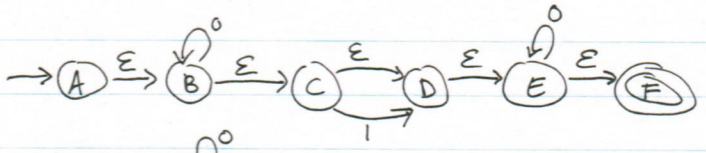


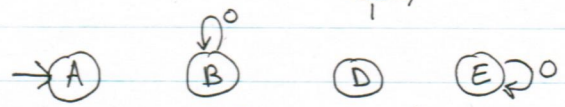
after 1
after 2.
after 3

to give us

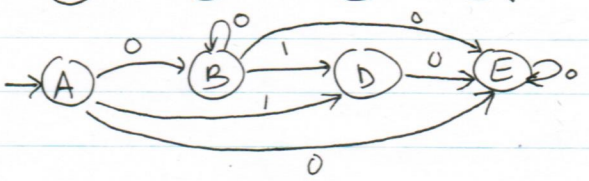


Ex.3

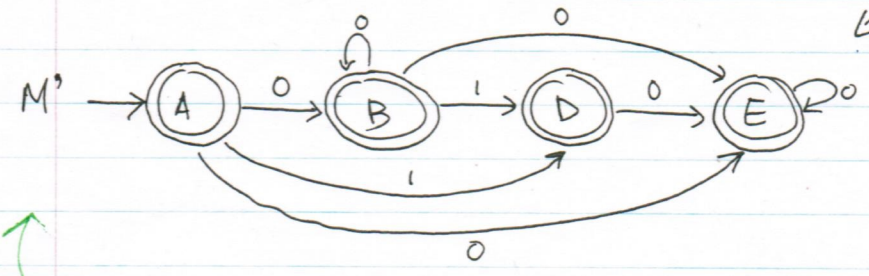
M 



← After 1

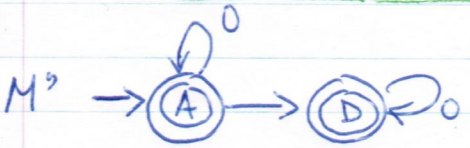


← After 2.



← After 3

↳ Any state can reach F so all are accepting states!



another way to write it (very condensed.)

↳ Thanks, Leif!