NFA , ε-NFA and Moore , Mealy machines

Lecture 9 and 10

Language of an NFA

- An NFA accepts *w* if *there exists at least one* path from the start state to an accepting (or final) state that is labeled by *w*
- $L(N) = \{ w \mid \hat{\delta}(q_0,w) \cap F \neq \Phi \}$

Advantages & Caveats for NFA

- Great for modeling regular expressions
  - String processing - e.g., grep, lexical analyzer
- Could a non-deterministic state machine be implemented in practice?
  - Probabilistic models could be viewed as extensions of non-deterministic state machines
    (e.g., toss of a coin, a roll of dice)
    - They are not the same though
  - A parallel computer could exist in multiple “states” at the same time

Technologies for NFAs

- Micron’s Automata Processor (introduced in 2013)
- 2D array of MISD (multiple instruction single data) fabric w/ thousands to millions of processing elements.
- 1 input symbol = fed to all states (i.e., cores)
- Non-determinism using circuits
- http://www.micronautomata.com/

But, DFAs and NFAs are equivalent in their power to capture langauges !!

Differences: DFA vs. NFA

- **DFA**

1. All transitions are deterministic
   - Each transition leads to exactly one state
2. For each state, transition on all possible symbols (alphabet) should be defined
3. Accepts input if the last state visited is in F
4. Sometimes harder to construct because of the number of states
5. Practical implementation is feasible

- **NFA**

1. Some transitions could be non-deterministic
   - A transition could lead to a subset of states
2. Not all symbol transitions need to be defined explicitly (if undefined will go to an error state – this is just a design convenience, not to be confused with "non-determinism")
3. Accepts input if *one of* the last states is in F
4. Generally easier than a DFA to construct
5. Practical implementations limited but emerging (e.g., Micron automata processor)

Equivalence of DFA & NFA

- Theorem:
  - A language L is accepted by a DFA *if and only if* it is accepted by an NFA.

Should be true for any L

- Proof:

1. If part:
   - Prove by showing every NFA can be converted to an equivalent DFA (in the next few slides…)

2. Only-if part is trivial:
   - Every DFA is a special case of an NFA where each state has exactly one transition for every input symbol. Therefore, if L is accepted by a DFA, it is accepted by a corresponding NFA. □

Proof for the if-part

- If-part: A language L is accepted by a DFA if it is accepted by an NFA
- rephrasing…
- Given any NFA N, we can construct a DFA D such that L(N)=L(D)

- How to convert an NFA into a DFA?
  - Observation: In an NFA, each transition maps to a *subset* of states
  - Idea: Represent:

    each "subset of NFA_states" → a single "DFA_state"

Subset construction

NFA to DFA by subset construction

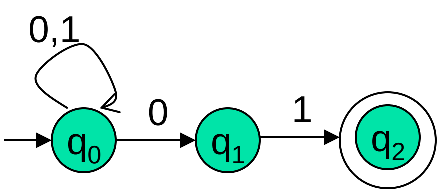
- Let $N = \{Q_N, \sum, \delta_N, q_0, F_N\}$
- <u>Goal:</u> Build $D=\{Q_D, \sum, \delta_D, \{q_0\}, F_D\}$ s.t. $L(D)=L(N)$
- <u>Construction:</u>
  1. Q_D= all subsets of Q_N (i.e., power set)
  2. F_D=set of subsets S of Q_N s.t. $S \cap F_N \neq \Phi$
  3. δ_D: for each subset S of Q_N and for each input symbol a in $\sum$:
     - $\delta_D(S,a) = \bigcup_{p \text{ in } s} \delta_N(p,a)$

Idea: To avoid enumerating all of power set, do "lazy creation of states"

NFA to DFA construction: Example

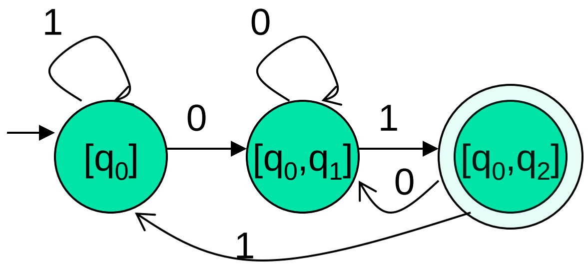
- *L = {w | w ends in 01}*

NFA:



δ_N	0	1
→ q_0	$\{q_0,q_1\}$	$\{q_0\}$
q_1	Ø	$\{q_2\}$
$*q_2$	Ø	Ø

DFA:



δ_D
~~Ø~~
→ $[q_0]$
~~$[q_1]$~~
~~$*[q_2]$~~
$[q_0,q_1]$
$*[q_0,q_2]$
~~$*[q_1,q_2]$~~
~~$*[q_0,q_1,q_2]$~~

δ_D	0	1
→ $[q_0]$	$[q_0,q_1]$	$[q_0]$
$[q_0,q_1]$	$[q_0,q_1]$	$[q_0,q_2]$
$*[q_0,q_2]$	$[q_0,q_1]$	$[q_0]$

0. Enumerate all possible subsets
1. Determine transitions
2. Retain only those states reachable from $\{q_0\}$

Correctness of subset construction

<u>Theorem:</u> If D is the DFA constructed from NFA N by subset construction, then L(D)=L(N)

- <u>Proof:</u>
  - Show that $\hat{\delta}_D(\{q_0\},w) \equiv \hat{\delta}_N(q_0,w\}$, for all w
  - Using induction on w's length:
    - Let w = xa
    - $\hat{\delta}_D(\{q_0\},xa) \equiv \delta_D(\hat{\delta}_N(q_0,x\}, a) \equiv \hat{\delta}_N(q_0,w\}$

Applications

- Text indexing
  - inverted indexing
  - For each unique word in the database, store all locations that contain it using an NFA or a DFA
- Find pattern P in text T
  - Example: Google querying
- Extensions of this idea:
  - PATRICIA tree, suffix tree

A few subtle properties of DFAs and NFAs

- The machine never really terminates.
  - It is always waiting for the next input symbol or making transitions.
- The machine decides when to <u>consume</u> the next symbol from the input and when to <u>ignore</u> it.
  - (but the machine can never <u>skip</u> a symbol)
- => A transition can happen even *without* really consuming an input symbol (think of consuming ε as a free token) – if this happens, then it becomes an ε-NFA (see next few slides).
- A single transition *cannot* consume more than one (non-ε) symbol.

FA with ε-Transitions

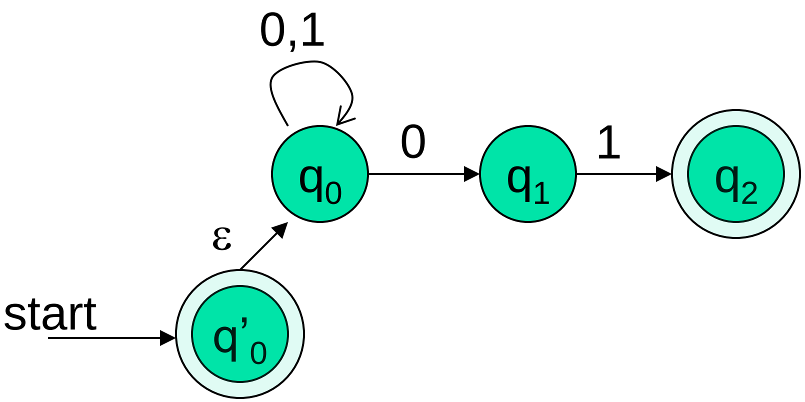
- We can allow <u>explicit</u> ε-transitions in finite automata
  - i.e., a transition from one state to another state without consuming any additional input symbol
  - Explicit ε-transitions between different states introduce non-determinism.
  - Makes it easier sometimes to construct NFAs

***<u>Definition:</u>* ε-NFAs are those NFAs with at least one explicit ε-transition defined.**

- ε -NFAs have one more column in their transition table

Example of an ε-NFA

L = {w | w is empty, <u>or</u> if non-empty will end in 01}



δ_E	0	1	ε
$\rightarrow$ $*q'_0$	{ }	{ }	$\{q'_0, q_0\}$
q_0	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0\}$
q_1	{ }	$\{q_2\}$	$\{q_1\}$
$*q_2$	{ }	{ }	$\{q_2\}$

$ECLOSE(q'_0)$

$ECLOSE(q_0)$

$ECLOSE(q_1)$

$ECLOSE(q_2)$

- ε-closure of a state q, ***ECLOSE(q)***, is the set of all states (including itself) that can be *reached* from q by repeatedly making an arbitrary number of ε-transitions.

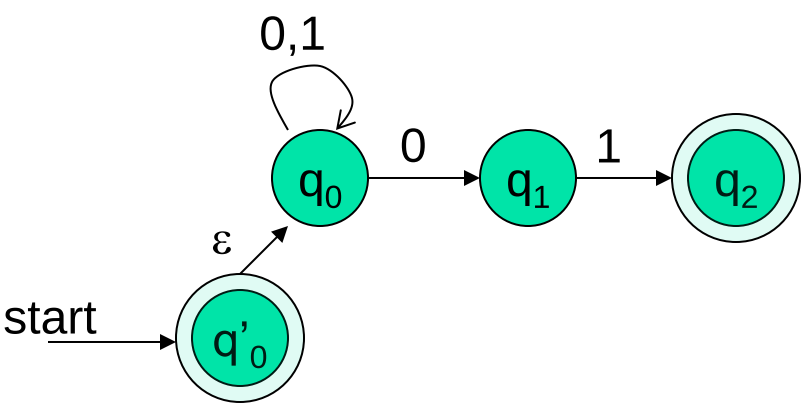
To simulate any transition:

Step 1) Go to all immediate destination states.

Step 2) From there go to all their ε-closure states as well.

Example of an ε-NFA

L = {w | w is empty, or if non-empty will end in 01}

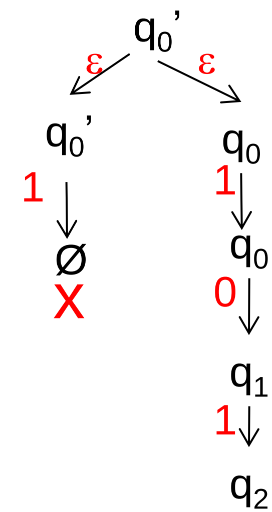


δ_E	0	1	ε
$*q'_0$	{ }	{ }	$\{q'_0, q_0\}$
q_0	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0\}$
q_1	{ }	$\{q_2\}$	$\{q_1\}$
$*q_2$	{ }	{ }	$\{q_2\}$

ECLOSE(q'_0)

ECLOSE(q_0)

Simulate for w=101:

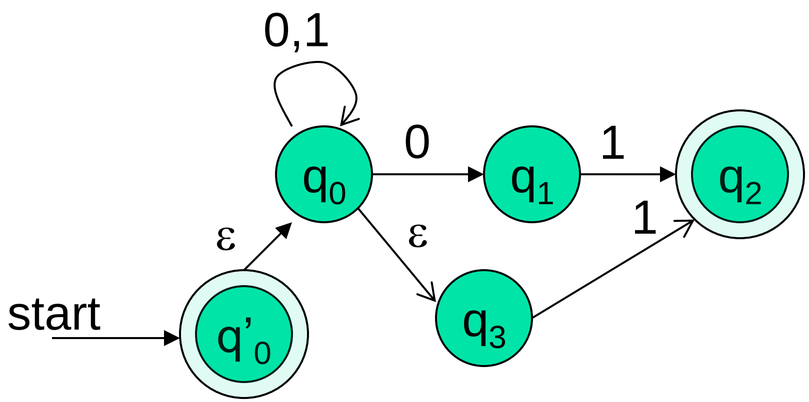


To simulate any transition:

Step 1) Go to all immediate destination states.

Step 2) From there go to all their ε-closure states as well.

Example of another ε-NFA



δ_E	0	1	ε
→*q'_0	{ }	{ }	$\{q'_0, q_0, q_3\}$
q_0	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0, q_3\}$
q_1	{ }	$\{q_2\}$	$\{q_1\}$
*q_2	{ }	{ }	$\{q_2\}$
q_3	{ }	$\{q_2\}$	$\{q_3\}$

<u>Simulate for w=101:</u>

?

Equivalency of DFA, NFA, ε-NFA

- Theorem: A language L is accepted by some ε-NFA if and only if L is accepted by some DFA

- Implication:
  - DFA ≡ NFA ≡ ε-NFA
  - (all accept Regular Languages)

Eliminating ε-transitions

Let $E = \{Q_E, \Sigma, \delta_E, q_0, F_E\}$ be an ε-NFA

Goal: To build DFA $D = \{Q_D, \Sigma, \delta_D, \{q_D\}, F_D\}$ s.t. $L(D)=L(E)$

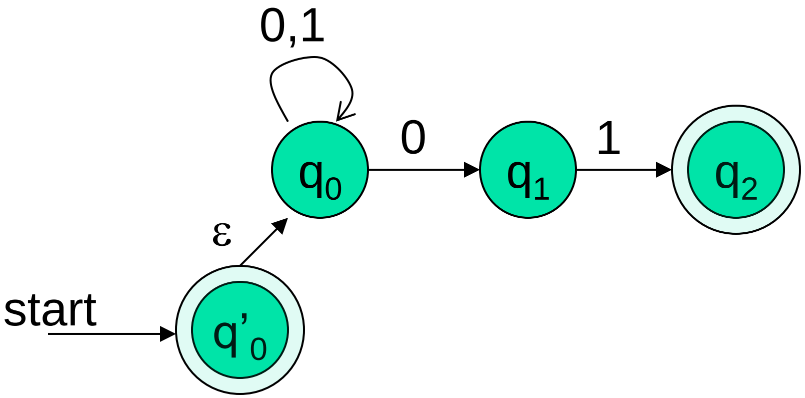
Construction:

1. Q_D= all reachable subsets of Q_E factoring in ε-closures
2. $q_D = \text{ECLOSE}(q_0)$
3. F_D=subsets S in Q_D s.t. $S \cap F_E \neq \Phi$
4. δ_D: for each subset S of Q_E and for each input symbol $a \in \Sigma$:
   - Let $R = \bigcup_{p \text{ in } s} \delta_E(p,a)$ // go to destination states
   - $\delta_D(S,a) = \bigcup_{r \text{ in } R} \text{ECLOSE}(r)$ // from there, take a union of all their ε-closures

Reading: Section 2.5.5 in book

Example: ε-NFA → DFA

L = {w | w is empty, or if non-empty will end in 01}

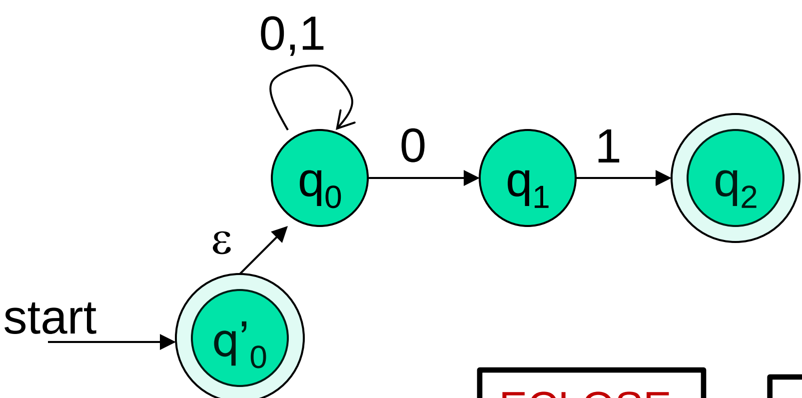


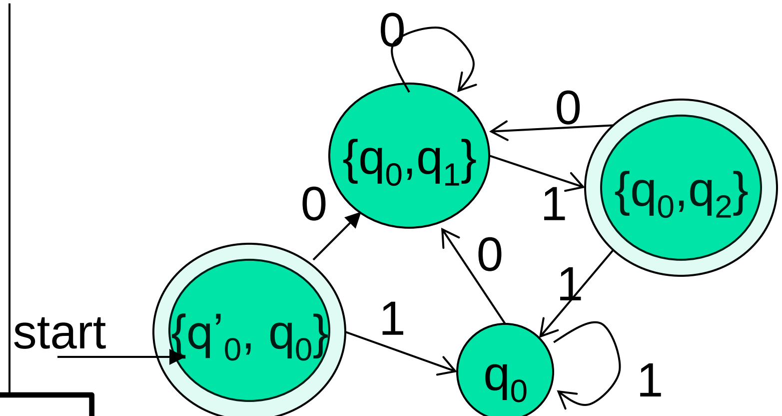
δ_E	0	1	ε
→ $*q'_0$	{ }	{ }	$\{q'_0,q_0\}$
q_0	$\{q_0,q_1\}$	$\{q_0\}$	$\{q_0\}$
q_1	{ }	$\{q_2\}$	$\{q_1\}$
$*q_2$	{ }	{ }	$\{q_2\}$

δ_D	0	1
→ $*\{q'_0,q_0\}$		
…		

Example: ε-NFA → DFA

L = {w | w is empty, or if non-empty will end in 01}





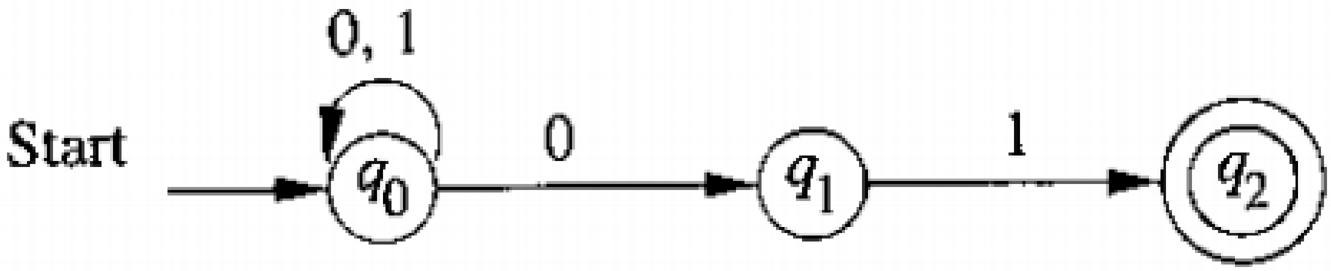
ECLOSE

union

δ_E	0	1	ε
→ *q'_0	{ }	{ }	$\{q'_0,q_0\}$
q_0	$\{q_0,q_1\}$	$\{q_0\}$	$\{q_0\}$
q_1	{ }	$\{q_2\}$	$\{q_1\}$
*q_2	{ }	{ }	$\{q_2\}$

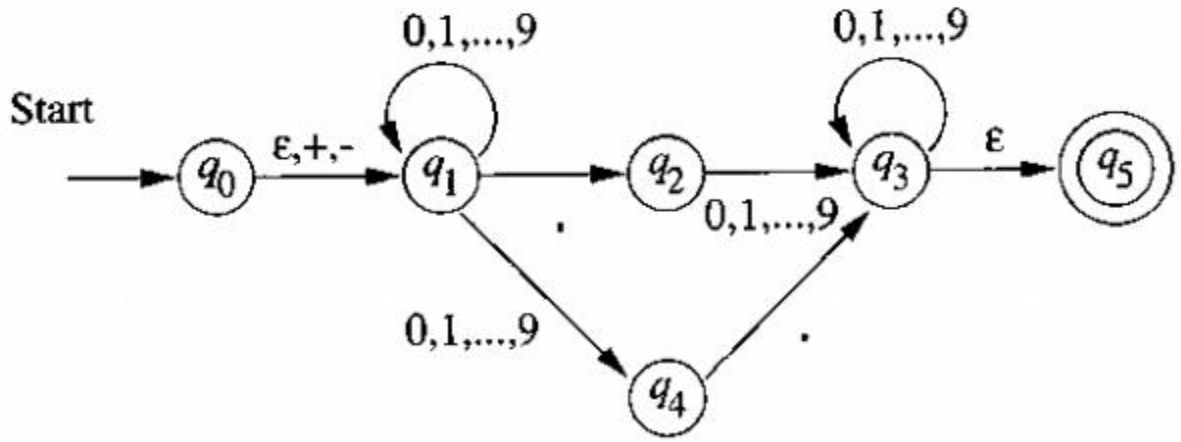
δ_D	0	1
→ *$\{q'_0,q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
$\{q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
*$\{q_0,q_2\}$	$\{q_0,q_1\}$	$\{q_0\}$

Extended Transition Function for NFAs



- For above NFA process string 00101??

Extended Function for ε-NFA



- Let compute $\delta(q0,5.6)$=??

Moore and Mealy Machines

Definition and example will be demonstrate in class

Summary

- DFA
  - Definition
  - Transition diagrams & tables
- Regular language
- NFA
  - Definition
  - Transition diagrams & tables
- DFA vs. NFA
- NFA to DFA conversion using subset construction
- Equivalency of DFA & NFA
- Removal of redundant states and including dead states
- ε-transitions in NFA
- Extended Transition function
- Moorey and Melay machine
- Text searching applications