Nondeterministic Finite Automata (NFA)

Dr. Mohammed Moshiul Hoque
CSE, CUET

Nondeterminism

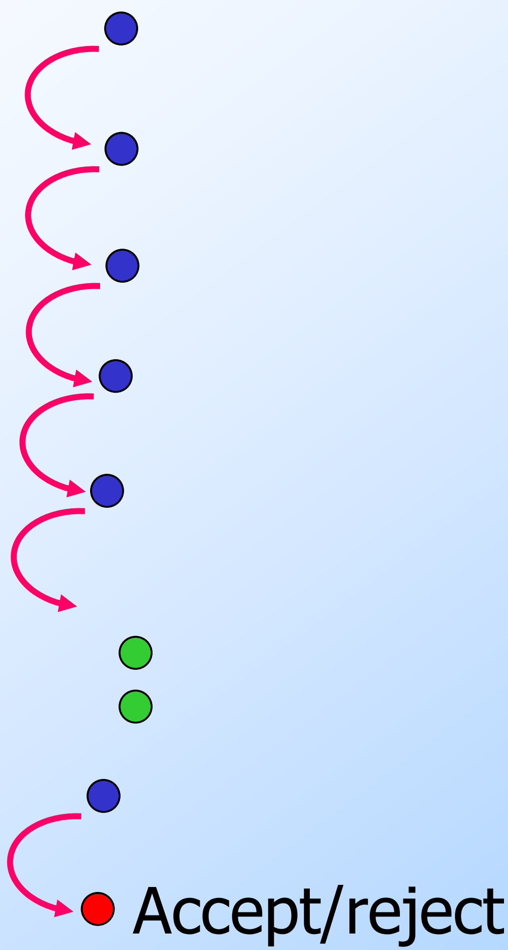
- A *nondeterministic finite automaton* has the ability to be in several states at once.
- Transitions from a state on an input symbol can be to any set of states.

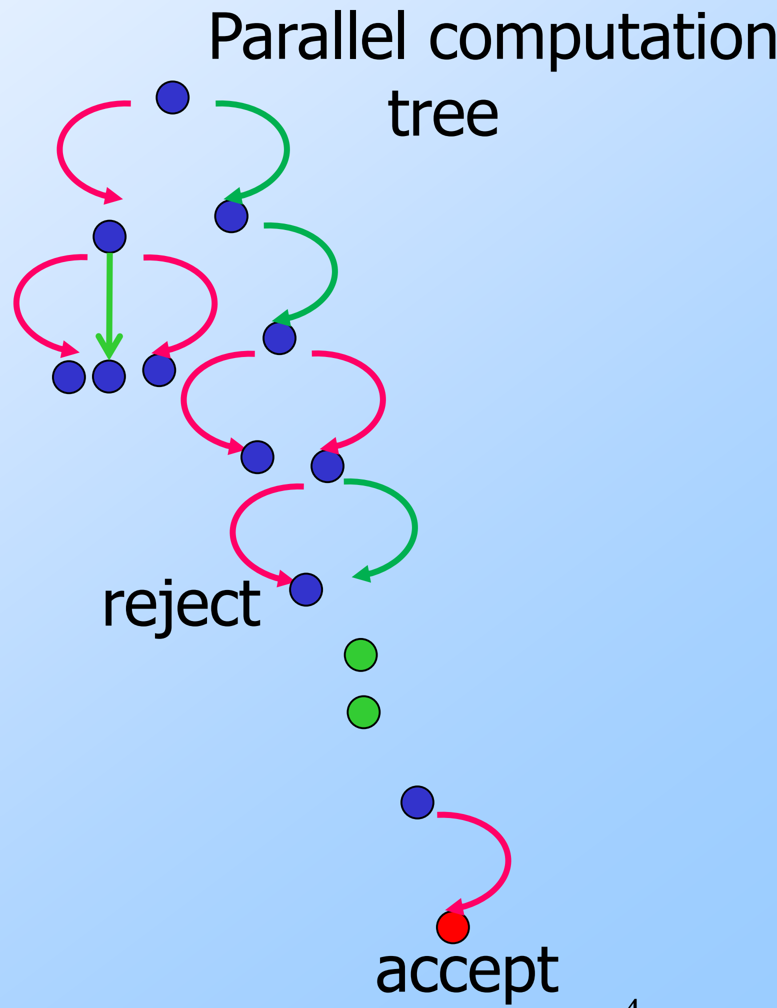
DFA vs. NFA

- DFA: δ returns a single state
- Every state of a DFA always has exactly one exiting transition arrow for each symbol in the alphabet
- Labels on the transition arrows are symbols from the alphabet

- NFA: δ returns a set of states
- Violets that rule
- In an NFA a state may have zero, one or many exiting arrows for each alphabet symbol
- NFA has an arrow with label $\in$
- NFA may have arrows labeled with members of alphabet/$\in$.
- Zero, one, or many arrows may exit from each state with label $\in$

DFA vs. NFA





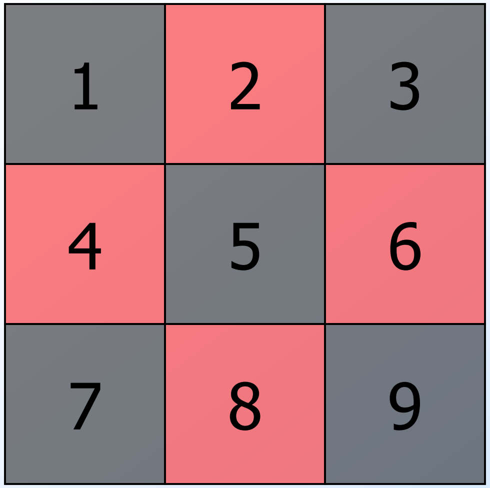
Nondeterminism – (2)

- Start in one start state.
- Accept if any sequence of choices leads to a final state.
- Intuitively: the NFA always "guesses right."

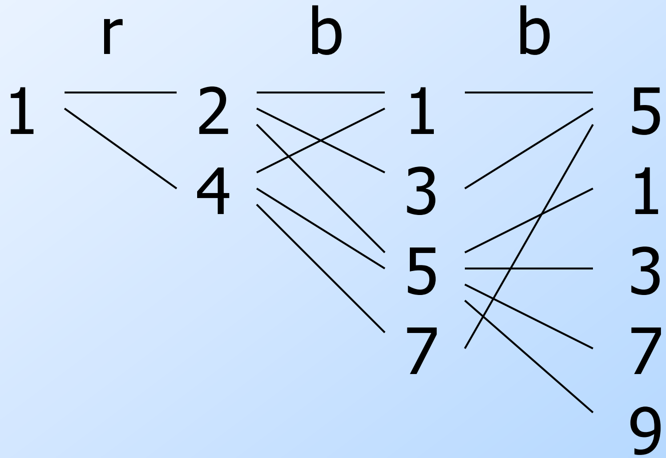
Example: Moves on a Chessboard

- States = squares.
- Inputs = r (move to an adjacent red square) and b (move to an adjacent black square).
- Start state, final state are in opposite corners.

Example: Chessboard – (2)



		r	b
→	1	2,4	5
	2	4,6	1,3,5
	3	2,6	5
	4	2,8	1,5,7
	5	2,4,6,8	1,3,7,9
	6	2,8	3,5,9
	7	4,8	5
	8	4,6	5,7,9
*	9	6,8	5



← Accept, since final state reached

Formal NFA

- A NFA is a 5-tuple, A = $(Q, \Sigma, \delta, q_0, F)$

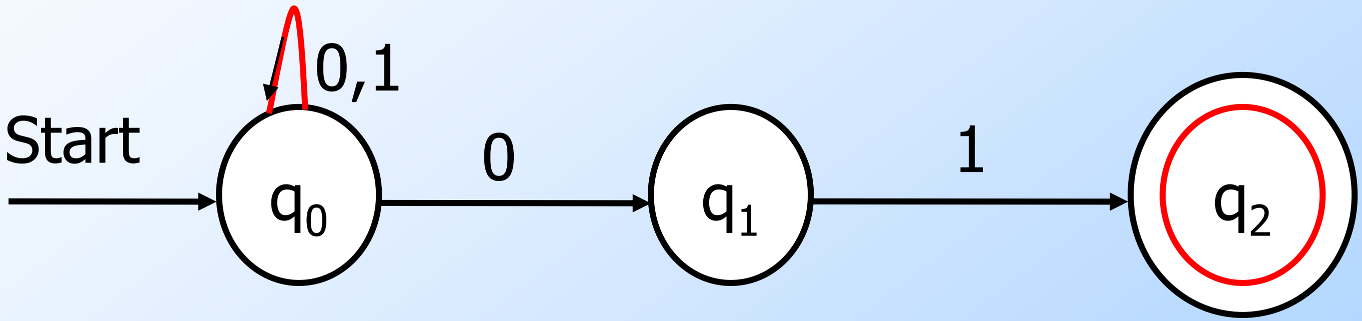
- A finite set of states, typically Q.
- An input alphabet, typically Σ.
- A transition function, typically δ.
- A start state in Q, typically q_0.
- A set of final states $F \subseteq Q$.

Transition Function of an NFA

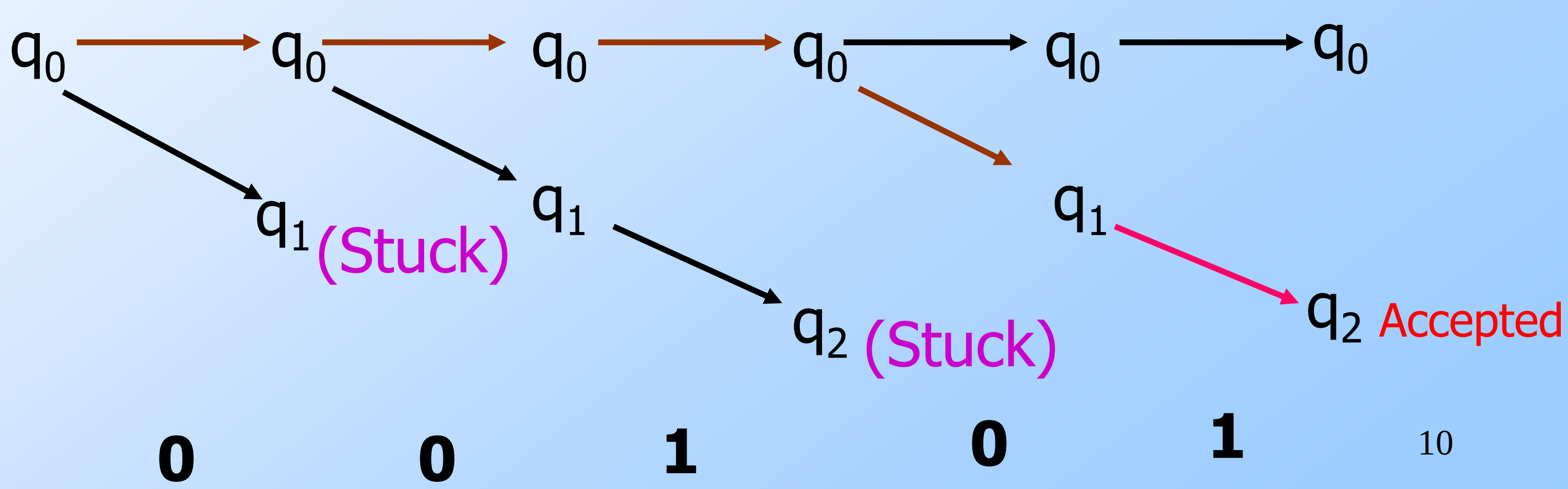
- The transition function is a function that takes a state in Q & an input symbol in Σ as arguments & returns a subset of Q.
- δ(q, a) is a set of states.
- Extend to strings as follows:
- Basis: $\delta(q, \epsilon) = \{q\}$
- Induction: $\delta(q, wa)$ = the union over all states p in $\delta(q, w)$ of $\delta(p, a)$

Example

- An NFA accepting all strings that end in 01

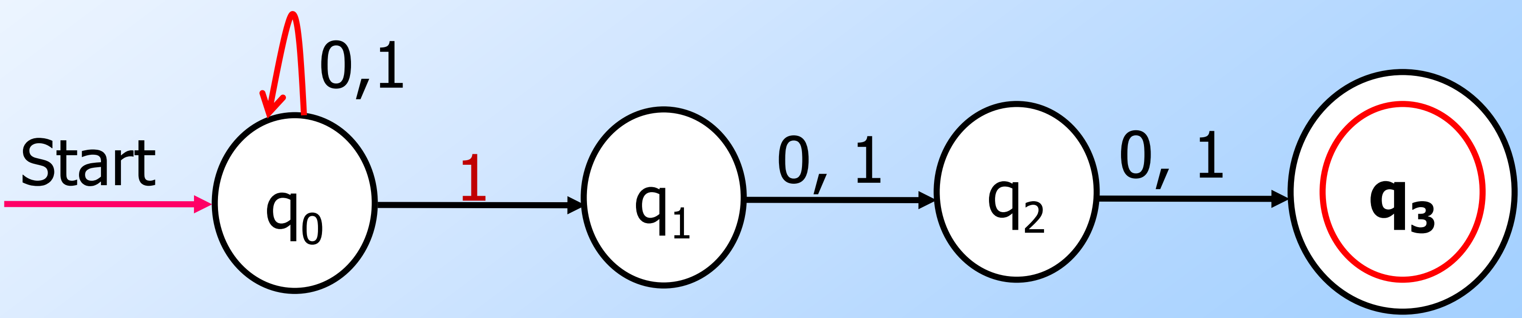


Input: **00101**



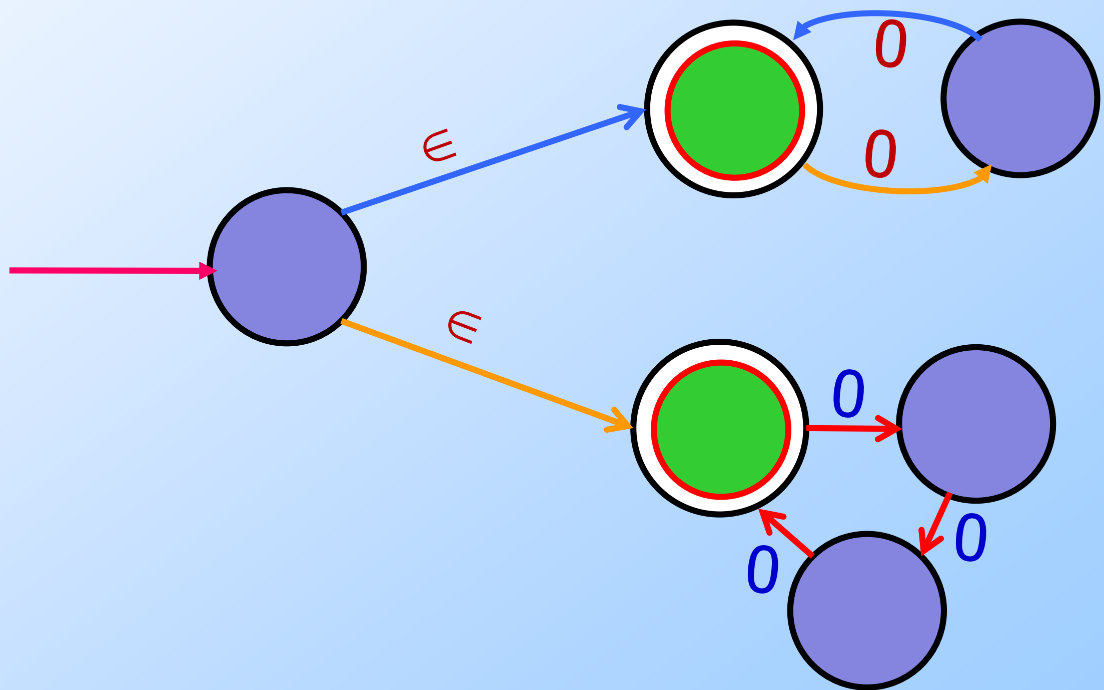
Example

- Let A be the language consisting of all strings over {0,1} containing a 1 in the third position from the end (000100 is in A but 0011 is not).



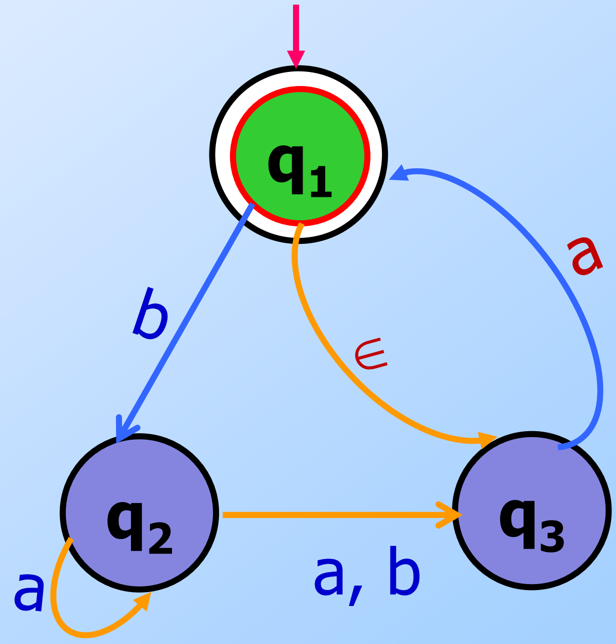
Example

- NFA that has an input alphabet {0} consisting of a single symbol. It accepts all strings of the form 0^k where k is a multiple of 2 or 3 (accept: $\in$, 00, 0000, 000000 but not 0, 00000)



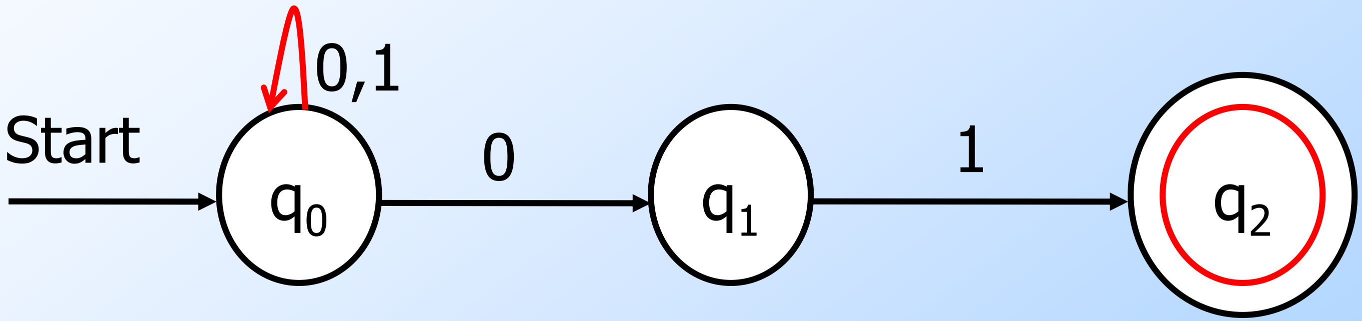
Example

Accept: $\in$, a, baba, baa
Reject: b, bb, babba



Transition Table

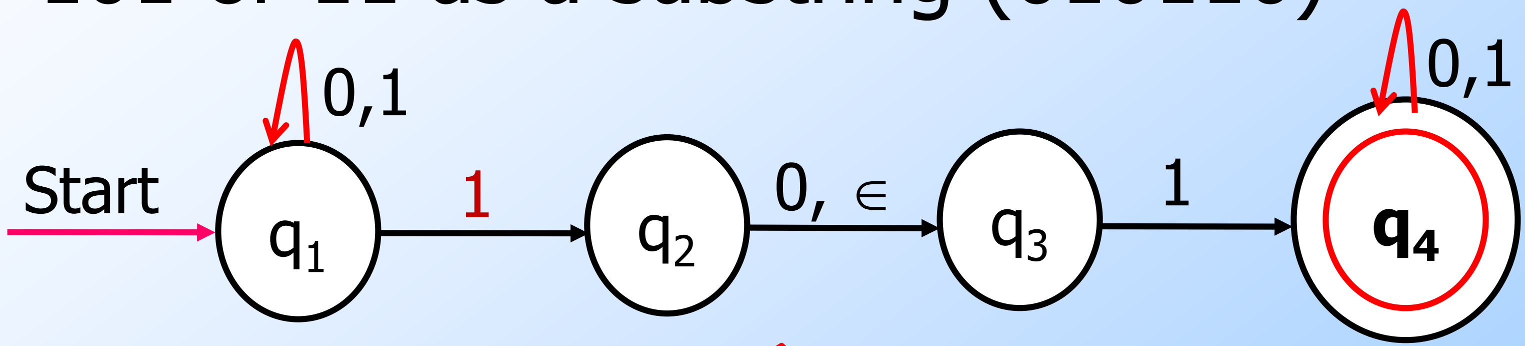
NFA A= $(\{q_0,q_1,q_2\},\{0,1\}, \delta ,q_0,\{q_2\})$



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

Transition Table

- Accept all strings that contains either 101 or 11 as a substring (010110)



1. $Q = \{q_1, q_2, q_3, q_4\}$
2. $\Sigma = \{0, 1\}$
3. **δ**

	0	**1**	**∈**
$\rightarrow q_1$	$\{q_1\}$	$\{q_1, q_2\}$	$\varnothing$
q_2	$\{q_3\}$	$\varnothing$	$\{q_3\}$
q_3	$\varnothing$	$\{q_4\}$	$\varnothing$
$*q_4$	$\{q_4\}$	$\{q_4\}$	$\varnothing$

4. Start state: q_1
5. $F = \{q_4\}$

Language of an NFA

- A string w is accepted by an NFA if $\delta(q_0, w)$ contains at least one final state.
- The language of the NFA is the set of strings it accepts.

Example: Language of an NFA

1	2	3
4	5	6
7	8	9

- For our chessboard NFA we saw that rbb is accepted.
- If the input consists of only b's, the set of accessible states alternates between {5} and {1,3,7,9}, so only even-length, nonempty strings of b's are accepted.
- What about strings with at least one r?

Equivalence of DFA's, NFA's

- A DFA can be turned into an NFA that accepts the same language.
- If $\delta_D(q, a) = p$, let the NFA have $\delta_N(q, a) = \{p\}$.
- Then the NFA is always in a set containing exactly one state – the state the DFA is in after reading the same input.
- Worst case: DFA-2^n states

  NFA- n states.

Equivalence – (2)

- Surprisingly, for any NFA there is a DFA that accepts the same language.
- Proof is the *subset construction*.
- The number of states of the DFA can be exponential in the number of states of the NFA.
- Thus, NFA's accept exactly the regular languages.

Subset Construction

- Given an NFA with states Q, inputs Σ, transition function δ_N, state state q_0, and final states F, construct equivalent DFA with:
  - States 2^Q (Set of subsets of Q).
  - Inputs Σ.
  - Start state $\{q_0\}$.
  - Final states = all those with a member of F.

Subset Construction

- Given, NFA: $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$
- Goal: DFA, $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$
- $L(D) = L(N)$

States

- Q_D is the set of subsets of Q_N
  - Q_D is the power set of Q_N
  - If Q_N has n states, Q_D will have 2^n states
- Inaccessible states can be thrown away, so effectively, the number of states D << 2^n

Subset construction

Final States

- F_D is the set of subsets S of Q_N such that $\mathbf{S} \cap \mathbf{F_N} \neq \varnothing$. That is F_D is all sets of N's states that include at least one accepting state of N.

Transition Function

- The transition function δ_D is defined by:

$\delta_D(\{q_1,\ldots,q_k\}, a)$ is the union over all $i = 1,\ldots,k$ of $\delta_N(q_i, a)$.

Critical Point

- The DFA states have *names* that are sets of NFA states.
- But as a DFA state, an expression like {p,q} must be read as a single symbol, not as a set.
- Analogy: a class of objects whose values are sets of objects of another class.

Subset Construction: Example 1

- Example: We'll construct the DFA equivalent of our "chessboard" NFA.

1	2	3
4	5	6
7	8	9

Example: Subset Construction

		r	b
→	1	2,4	5
	2	4,6	1,3,5
	3	2,6	5
	4	2,8	1,5,7
	5	2,4,6,8	1,3,7,9
	6	2,8	3,5,9
	7	4,8	5
	8	4,6	5,7,9
*	9	6,8	5

		r	b
→	{1}	{2,4}	{5}
	{2,4}		
	{5}		

Alert: What we're doing here is the *lazy* form of DFA construction, where we only construct a state if we are forced to.

Example: Subset Construction

		r	b
→	1	2,4	5
	2	4,6	1,3,5
	3	2,6	5
	4	2,8	1,5,7
	5	2,4,6,8	1,3,7,9
	6	2,8	3,5,9
	7	4,8	5
	8	4,6	5,7,9
*	9	6,8	5

		r	b
→	{1}	{2,4}	{5}
	{2,4}	{2,4,6,8}	{1,3,5,7}
	{5}		
	{2,4,6,8}		
	{1,3,5,7}		

Example: Subset Construction

		r	b
→	1	2,4	5
	2	4,6	1,3,5
	3	2,6	5
	4	2,8	1,5,7
	5	2,4,6,8	1,3,7,9
	6	2,8	3,5,9
	7	4,8	5
	8	4,6	5,7,9
*	9	6,8	5

		r	b
→	{1}	{2,4}	{5}
	{2,4}	{2,4,6,8}	{1,3,5,7}
	{5}	{2,4,6,8}	{1,3,7,9}
	{2,4,6,8}		
	{1,3,5,7}		
*	{1,3,7,9}		

Example: Subset Construction

		r	b
→	1	2,4	5
	2	4,6	1,3,5
	3	2,6	5
	4	2,8	1,5,7
	5	2,4,6,8	1,3,7,9
	6	2,8	3,5,9
	7	4,8	5
	8	4,6	5,7,9
*	9	6,8	5

		r	b
→	{1}	{2,4}	{5}
	{2,4}	{2,4,6,8}	{1,3,5,7}
	{5}	{2,4,6,8}	{1,3,7,9}
	{2,4,6,8}	{2,4,6,8}	{1,3,5,7,9}
	{1,3,5,7}		
*	{1,3,7,9}		
*	{1,3,5,7,9}		

Example: Subset Construction

		r	b
→	1	2,4	5
	2	4,6	1,3,5
	3	2,6	5
	4	2,8	1,5,7
	5	2,4,6,8	1,3,7,9
	6	2,8	3,5,9
	7	4,8	5
	8	4,6	5,7,9
*	9	6,8	5

		r	b
→	{1}	{2,4}	{5}
	{2,4}	{2,4,6,8}	{1,3,5,7}
	{5}	{2,4,6,8}	{1,3,7,9}
	{2,4,6,8}	{2,4,6,8}	{1,3,5,7,9}
	{1,3,5,7}	{2,4,6,8}	{1,3,5,7,9}
*	{1,3,7,9}		
*	{1,3,5,7,9}		

Example: Subset Construction

		r	b
→	1	2,4	5
	2	4,6	1,3,5
	3	2,6	5
	4	2,8	1,5,7
	5	2,4,6,8	1,3,7,9
	6	2,8	3,5,9
	7	4,8	5
	8	4,6	5,7,9
*	9	6,8	5

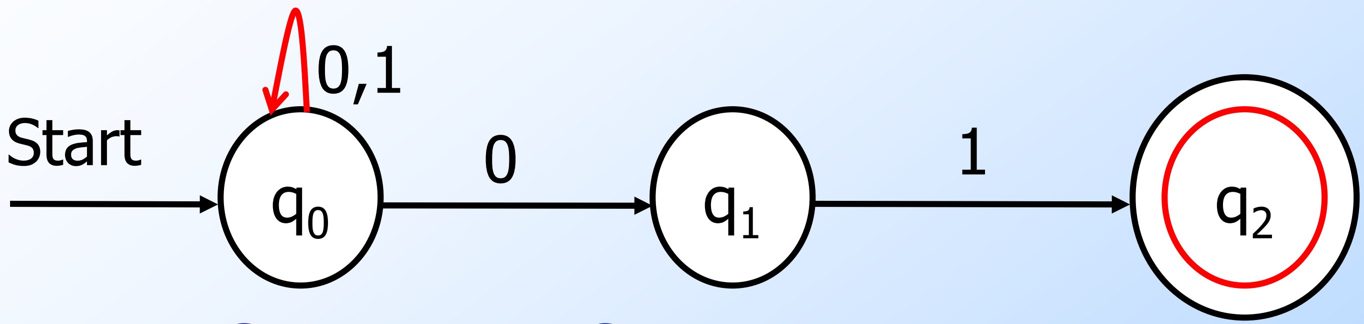
		r	b
→	{1}	{2,4}	{5}
	{2,4}	{2,4,6,8}	{1,3,5,7}
	{5}	{2,4,6,8}	{1,3,7,9}
	{2,4,6,8}	{2,4,6,8}	{1,3,5,7,9}
	{1,3,5,7}	{2,4,6,8}	{1,3,5,7,9}
*	{1,3,7,9}	{2,4,6,8}	{5}
*	{1,3,5,7,9}		

Example: Subset Construction

		r	b
→	1	2,4	5
	2	4,6	1,3,5
	3	2,6	5
	4	2,8	1,5,7
	5	2,4,6,8	1,3,7,9
	6	2,8	3,5,9
	7	4,8	5
	8	4,6	5,7,9
*	9	6,8	5

		r	b
→	{1}	{2,4}	{5}
	{2,4}	{2,4,6,8}	{1,3,5,7}
	{5}	{2,4,6,8}	{1,3,7,9}
	{2,4,6,8}	{2,4,6,8}	{1,3,5,7,9}
	{1,3,5,7}	{2,4,6,8}	{1,3,5,7,9}
*	{1,3,7,9}	{2,4,6,8}	{5}
*	{1,3,5,7,9}	{2,4,6,8}	{1,3,5,7,9}

Example 2



$\delta_D(\{q_0, q_2\}, 0) = \delta_N(\{q_0, 0) \cup \delta_N(\{q_2, 0) = \{q_0, q_1\} \cup \varnothing = \{q_0, q_1\}$

$\delta_D(\{q_0, q_2\}, 1) = \delta_N(\{q_0, 1) \cup \delta_N(\{q_2, 1) = \{q_0\} \cup \varnothing = \{q_0\}$

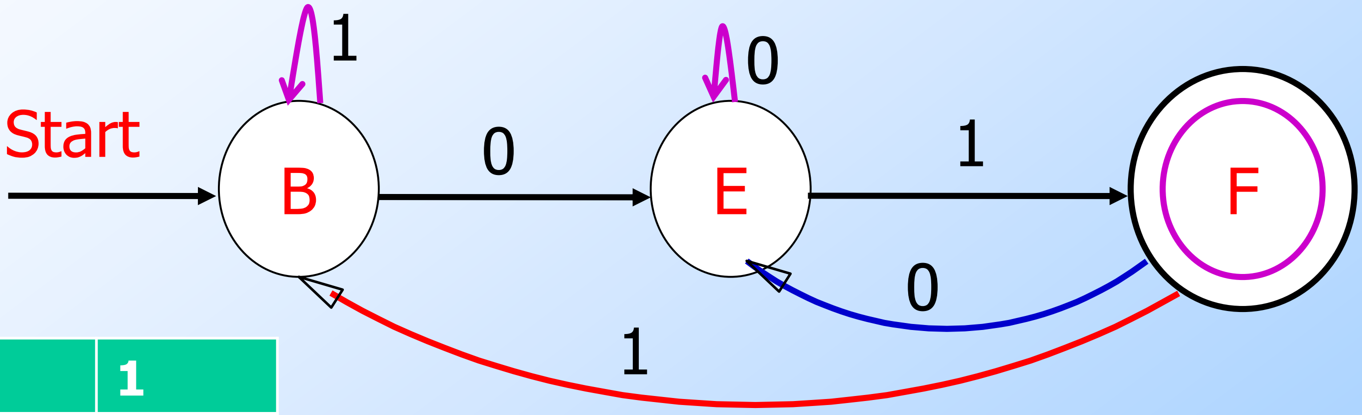
	0	1
$\varnothing$	$\varnothing$	$\varnothing$
$\rightarrow\{q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
$\{q_1\}$	$\varnothing$	$\{q_2\}$
$*\{q_2\}$	$\varnothing$	$\varnothing$
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
$*\{q_0,q_2\}$	$\{q_0,q_1\}$	$\{q_0\}$
$*\{q_1,q_2\}$	$\varnothing$	$\{q_2\}$
$*\{q_0,q_1,q_2\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$

Example 2

- NFA N Accepts all strings that end in 01
- N's set of states: $\{q_1, q_2, q_3\}$ =03
- Subset construction: DFA need $2^3 = 8$ states
- Assign new names: A for $\varnothing$, B for $\{q_0\}$

	0	1
A	A	A
→B	E	B
C	A	D
*D	A	A
E	E	F
*F	E	B
*G	A	D
*H	E	F

Example 2

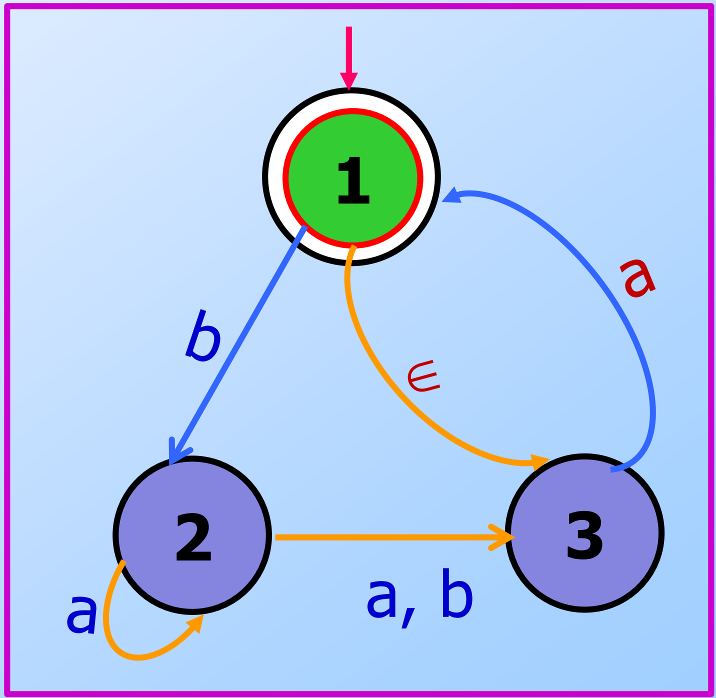


	0	1
A	A	A
→B	E	B
C	A	D
*D	A	A
E	E	F
*F	E	B
*G	A	D
*H	E	F

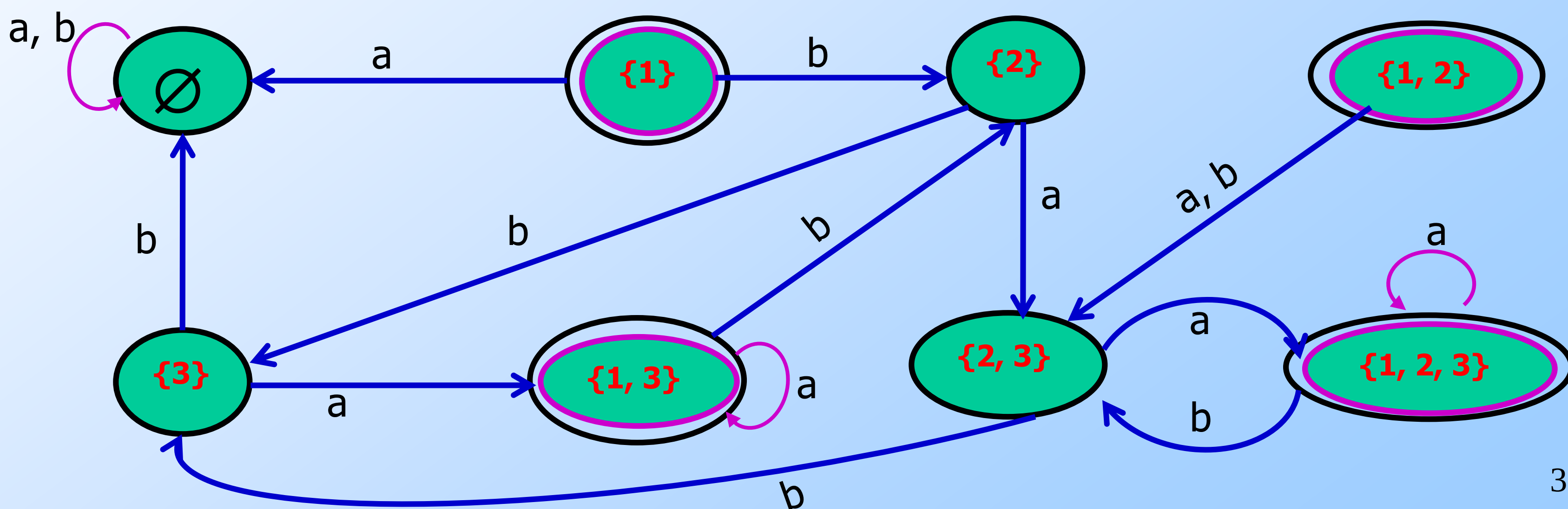
- From 08 states, starting in start state B, can only reach states B, E and F
- other 05 states are inaccessible from B

Example 3

- $N = (Q, \{a, b\}, \delta, 1, \{1\})$
- $Q = \{1, 2, 3\}$ = 03 states
- DFA states = 08
- $\{\varnothing, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$

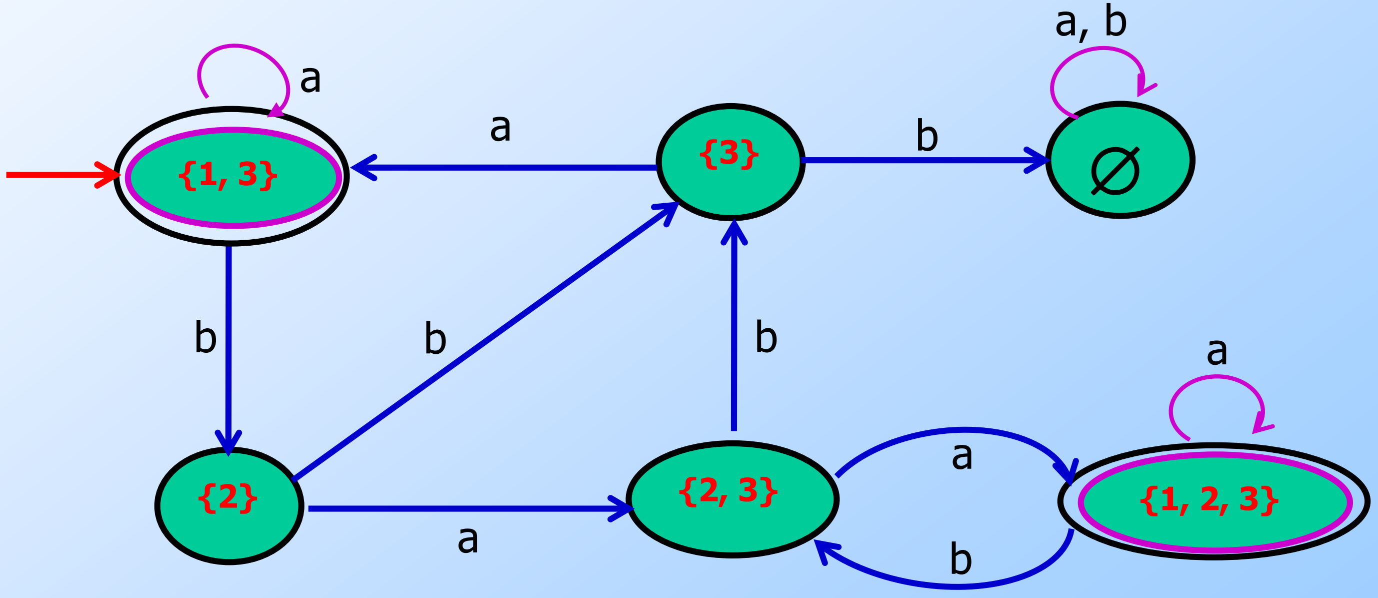


	a	b	ε
∅	∅	∅	∅
{1}	∅	{2}	{3}
{2}	{2, 3}	{3}	∅
{3}	{1, 3}	∅	∅
{1, 2}	{2, 3}	{2, 3}	∅
{1, 3}	{1, 3}	{2}	∅
{2, 3}	{1, 2, 3}	{3}	∅
{1, 2, 3}	{1, 2, 3}	{2, 3}	∅



Example 3

Simplified: no arrows point at states {1} & {1, 2}
May be removed without affecting the performance



Dead States & DFA's Missing Some Transitions

- A DFA to have a transition from any state, on any input symbol, to exactly one state.
- Sometimes, it is more convenient to design the DFA to "**die**" in situations where we know it is impossible for any extension of the input sequence to be accepted.
- If we use subset construction to convert a NFA to a DFA, the automation looks almost the same, but it includes a dead state.
- That is, a non-accepting state that goes to itself on every possible input symbol.
- Dead state: $\varnothing$ (empty set of states)

Dead States & DFA's Missing Some Transitions

- In general, we can add a dead state to any automaton that has *no more* than one transition for any state & input symbol.
- Then add a transition to the dead state from each other state q, on all input symbols for which q has no other transition.
- The result will be a DFA in the strict sense
- Thus, we shall sometimes refer to an automaton as a DFA if it has *at most* one transition out of any state on any symbol, rather than if it has *exactly one* transition

NFA's With ε-Transitions

- We can allow state-to-state transitions on ε input.
- These transitions are done spontaneously, without looking at the input string.
- A convenience at times, but still only regular languages are accepted.
- Useful in proving equivalence: finite automata = regular expressions

Example: Uses of ε-transitions

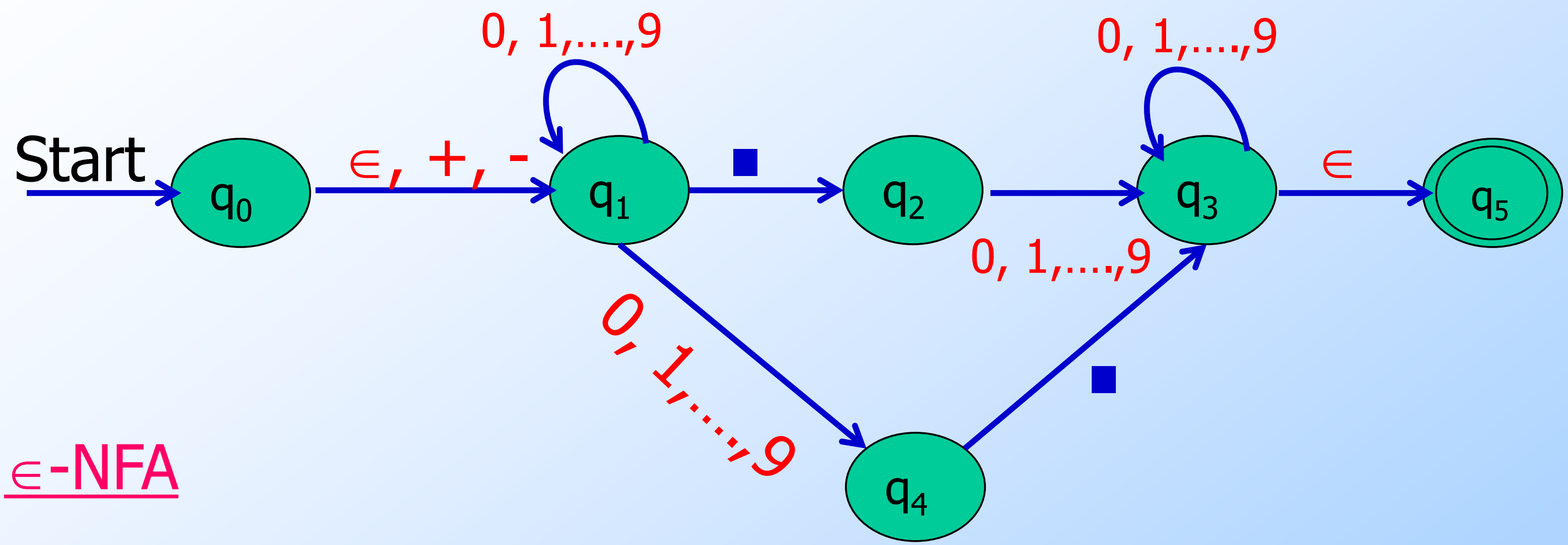
- ε-NFA that accepts decimal numbers consisting of:

-an optional +/- sign

-A string of digits

-Another string of digits. Either this string of digits or the string (2) can be empty, but at least one of the two strings of digits must be nonempty.

An $\in$-NFA Accepting decimal numbers



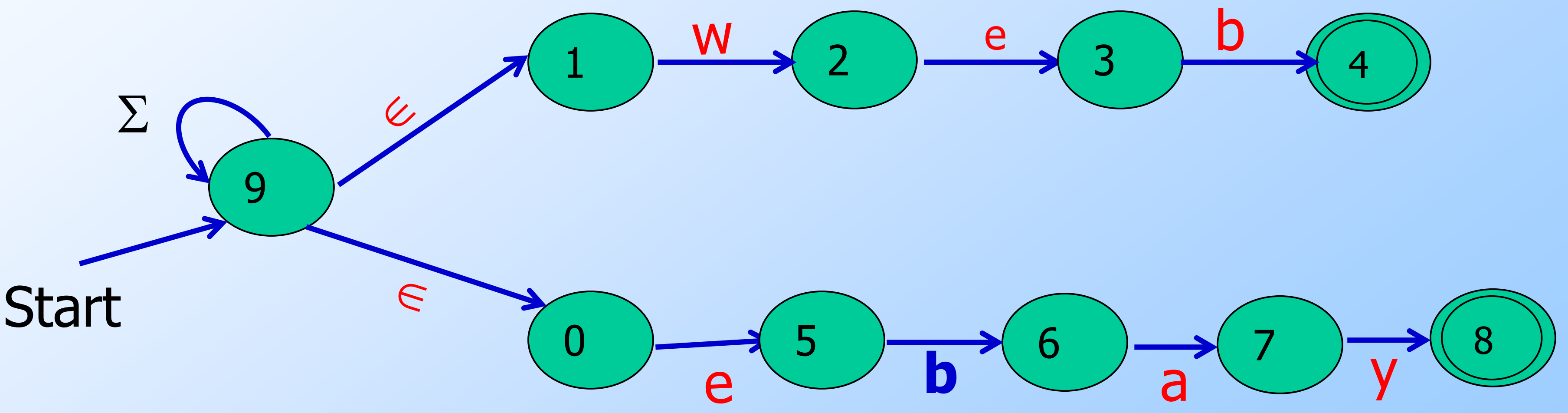
$\in$-NFA

$E = (\{q_0, q_1, q_2, q_4, q_5\}, \{., +, -, 0, 1, \ldots, 9\}, \delta, q_0, \{q_5\})$

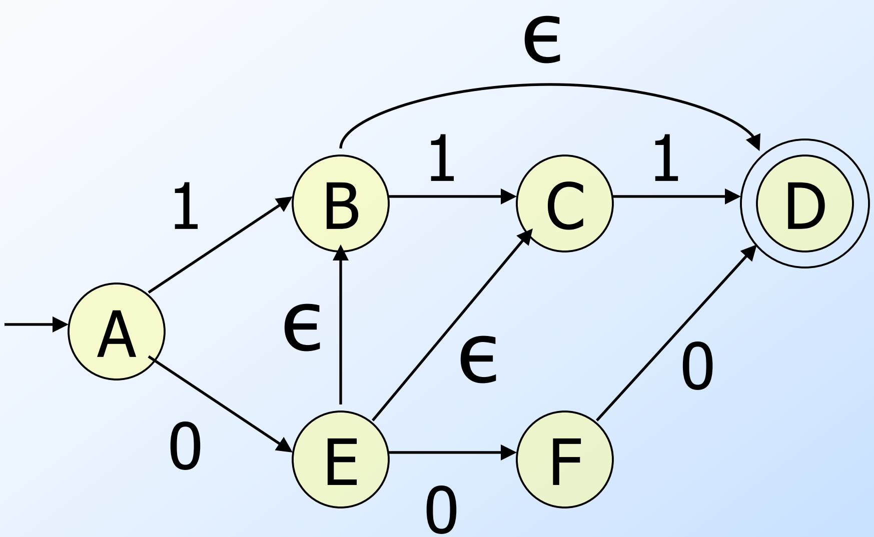
	$\in$	+, -	.	0, 1,…., 9
$\rightarrow q_0$	$\{q_1\}$	$\{q_1\}$	$\varnothing$	$\varnothing$
q_1	$\varnothing$	$\varnothing$	$\{q_2\}$	$\{q_1, q_4\}$
q_2	$\varnothing$	$\varnothing$	$\varnothing$	$\{q_3\}$
q_3	$\{q_5\}$	$\varnothing$	$\varnothing$	$\{q_3\}$
q_4	$\varnothing$	$\varnothing$	$\{q_3\}$	$\varnothing$
$*q_5$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

Example: Uses of ε-transitions

∈-transitions help to recognize keywords: **web, ebay**



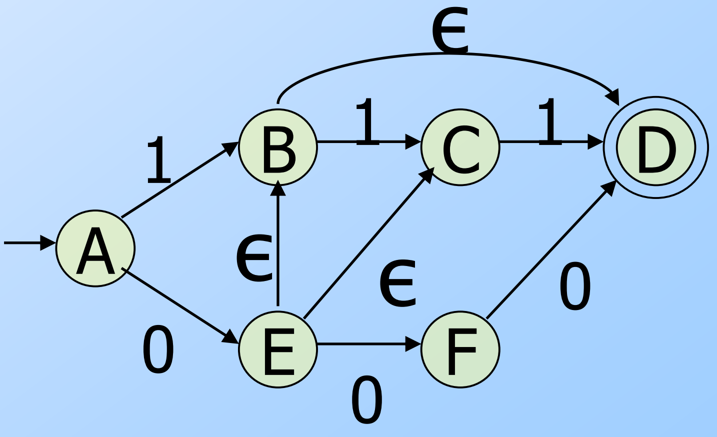
Example: ϵ-NFA



		0	1	ϵ
→	A	{E}	{B}	∅
	B	∅	{C}	{D}
	C	∅	{D}	∅
*	D	∅	∅	∅
	E	{F}	∅	{B, C}
	F	{D}	∅	∅

Closure of States

- CL(q) = set of states you can reach from state q following only arcs labeled ϵ.
- Example: CL(A) = {A};
  CL(E) = {B, C, D, E}.



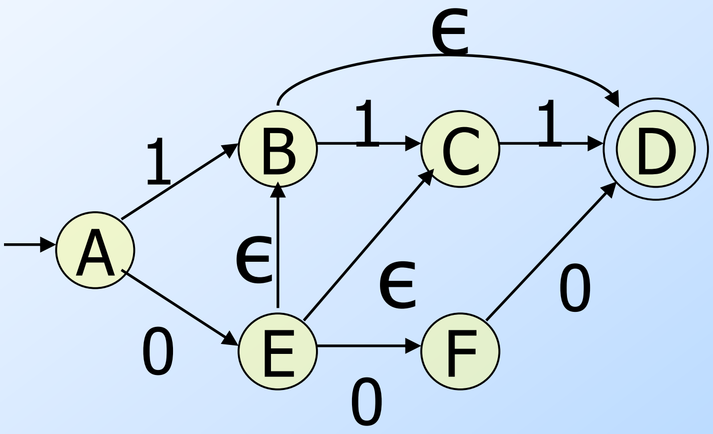
- Closure of a set of states = union of the closure of each state.

Extended Delta

- Basis: $\hat{\delta}(q, \epsilon) = CL(q)$.
- Induction: $\hat{\delta}(q, xa)$ is computed as follows:
  1. Start with $\hat{\delta}(q, x) = S$.
  2. Take the union of $CL(\delta(p, a))$ for all p in S.
- Intuition: $\hat{\delta}(q, w)$ is the set of states you can reach from q following a path labeled w.

And notice that $\delta(q, a)$ is *not* that set of states, for symbol a.

Example: Extended Delta



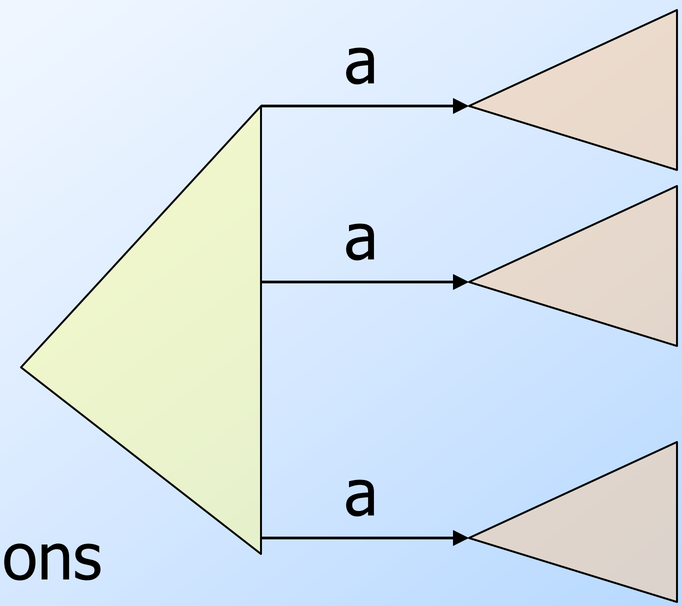
- $\hat{\delta}(A, \epsilon) = CL(A) = \{A\}$.
- $\hat{\delta}(A, 0) = CL(\{E\}) = \{B, C, D, E\}$.
- $\hat{\delta}(A, 01) = CL(\{C, D\}) = \{C, D\}$.
- *Language* of an ϵ-NFA is the set of strings w such that $\hat{\delta}(q_0, w)$ contains a final state.

Equivalence of NFA, ϵ-NFA

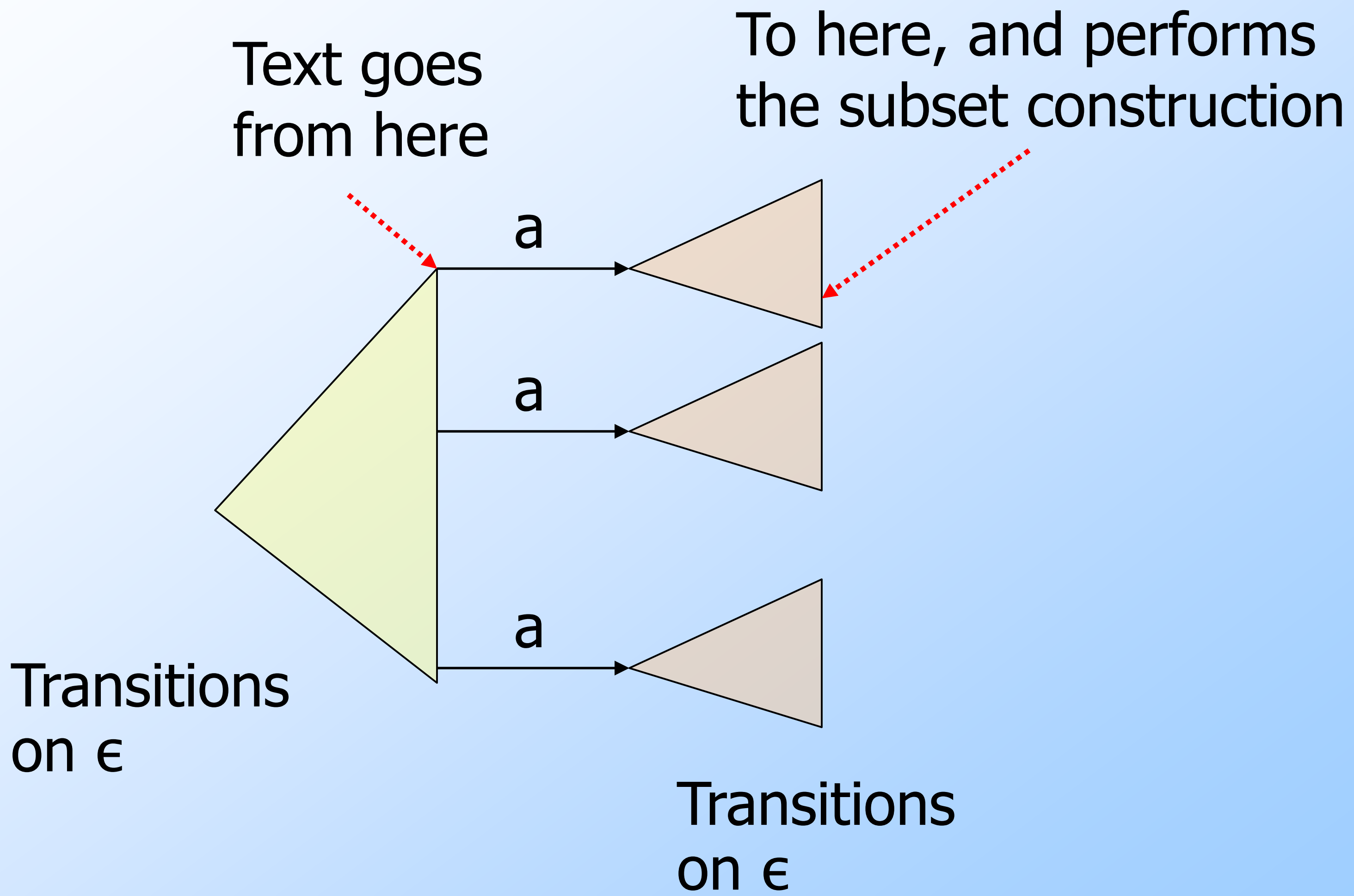
- Every NFA is an ϵ-NFA.
  - It just has no transitions on ϵ.
- Converse requires us to take an ϵ-NFA and construct an NFA that accepts the same language.
- We do so by combining ϵ–transitions with the next transition on a real input.

Warning: This treatment is a bit different from that in the text.

Picture of ϵ-Transition Removal



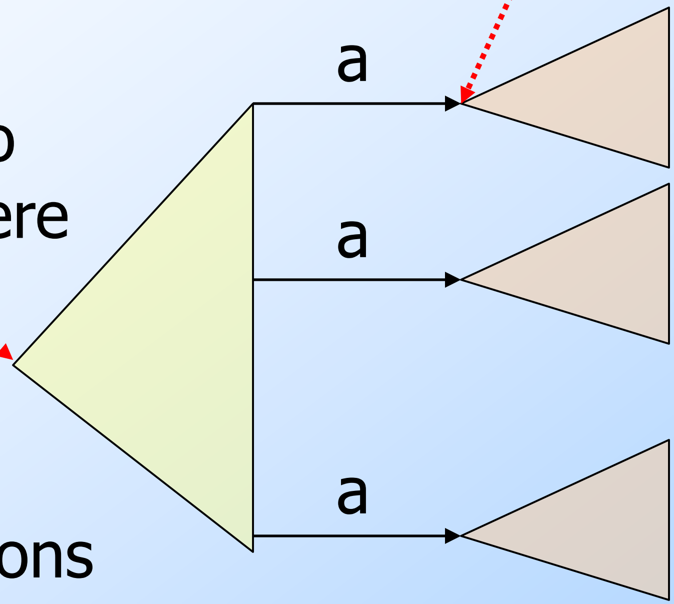
Picture of ϵ-Transition Removal



Picture of ϵ-Transition Removal

To here, with no subset construction

We'll go from here



Transitions on ϵ

Transitions on ϵ

Equivalence – (2)

- Start with an ϵ-NFA with states Q, inputs Σ, start state q_0, final states F, and transition function δ_E.
- Construct an "ordinary" NFA with states Q, inputs Σ, start state q_0, final states F', and transition function δ_N.

Equivalence – (3)

- Compute $\delta_N(q, a)$ as follows:

1. Let S = CL(q).
2. $\delta_N(q, a)$ is the union over all p in S of $\delta_E(p, a)$.

- F' = the set of states q such that CL(q) contains a state of F.
- Intuition: δ_N incorporates ϵ–transitions before using *a* but not after.

Equivalence – (4)

- Prove by induction on |w| that

  $CL(\delta_N(q_0, w)) = \hat{\delta}_E(q_0, w)$.

- Thus, the ϵ-NFA accepts w if and only if the "ordinary" NFA does.

Example: ϵ-NFA-to-NFA

Interesting closures: CL(B) = {B,D}; CL(E) = {B,C,D,E}

		0	1	ϵ
→	A	{E}	{B}	∅
	B	∅	{C}	{D}
	C	∅	{D}	∅
*	D	∅	∅	∅
	E	{F}	∅	{B, C}
	F	{D}	∅	∅

ϵ-NFA

		0	1
→	A	{E}	{B}
*	B	∅	{C}
	C	∅	{D}
*	D	∅	∅
*	E	{F}	{C, D}
	F	{D}	∅

Since closures of B and E include final state D.

Since closure of E includes B and C; which have transitions on 1 to C and D.

Summary

- DFA's, NFA's, and ϵ–NFA's all accept exactly the same set of languages: the regular languages.
- The NFA types are easier to design and may have exponentially fewer states than a DFA.
- But only a DFA can be implemented!