BILL, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!

Nondeterministic Finite Automata (NFA)

An Interesting Example of a DFA

With neighbor find DFA's for the following. Note numb. states.
$\Sigma^* a$
$\Sigma^* a \Sigma$
$\Sigma^* a \Sigma^2$

$\Sigma^* a \Sigma^2$

https://www.cs.umd.edu/users/gasarch/COURSES/452/S21/notes/dfa3.JPG

$\Sigma^* a \Sigma^2$

https://www.cs.umd.edu/users/gasarch/COURSES/452/S21/notes/dfa3.JPG

The number of states is 8.

$\Sigma^* a \Sigma^2$

https://www.cs.umd.edu/users/gasarch/COURSES/452/S21/notes/dfa3.JPG
The number of states is 8.

More generally:
$\Sigma^* a \Sigma^i$ can be done with 2^{i+1} states.

$\Sigma^* a \Sigma^2$

https://www.cs.umd.edu/users/gasarch/COURSES/452/S21/notes/dfa3.JPG

The number of states is 8.

More generally:

$\Sigma^* a \Sigma^i$ can be done with 2^{i+1} states.

Prove for $\Sigma^* a \Sigma^3$, with a table.

$\Sigma^* a \Sigma^2$

https://www.cs.umd.edu/users/gasarch/COURSES/452/S21/notes/dfa3.JPG

The number of states is 8.

More generally:

$\Sigma^* a \Sigma^i$ can be done with 2^{i+1} states.

Prove for $\Sigma^* a \Sigma^3$, with a table.

Might be on $2^{\{\text{HW, MIDTERM, FINAL}\}}$.

$\Sigma^* a \Sigma^2$

https://www.cs.umd.edu/users/gasarch/COURSES/452/S21/notes/dfa3.JPG

The number of states is 8.

More generally:

$\Sigma^* a \Sigma^i$ can be done with 2^{i+1} states.

Prove for $\Sigma^* a \Sigma^3$, with a table.

Might be on $2^{\{\text{HW, MIDTERM, FINAL}\}}$.

8 possibilities.

$\Sigma^* a \Sigma^2$

https://www.cs.umd.edu/users/gasarch/COURSES/452/S21/notes/dfa3.JPG

The number of states is 8.

More generally:

$\Sigma^* a \Sigma^i$ can be done with 2^{i+1} states.

Prove for $\Sigma^* a \Sigma^3$, with a table.

Might be on $2^{\{\text{HW, MIDTERM, FINAL}\}}$.

8 possibilities.

Is there a smaller DFA for $\Sigma^* a \Sigma^i$? Fewer than 2^{i+1} states?

$\Sigma^* a \Sigma^2$

`https://www.cs.umd.edu/users/gasarch/COURSES/452/S21/notes/dfa3.JPG`

The number of states is 8.

More generally:
$\Sigma^* a \Sigma^i$ can be done with 2^{i+1} states.

Prove for $\Sigma^* a \Sigma^3$, with a table.

Might be on $2^{\{\text{HW, MIDTERM, FINAL}\}}$.

8 possibilities.

Is there a smaller DFA for $\Sigma^* a \Sigma^i$? Fewer than 2^{i+1} states? No.
We may prove this later.

$\Sigma^* a \Sigma^2$

`https://www.cs.umd.edu/users/gasarch/COURSES/452/S21/notes/dfa3.JPG`

The number of states is 8.

More generally:
$\Sigma^* a \Sigma^i$ can be done with 2^{i+1} states.

Prove for $\Sigma^* a \Sigma^3$, with a table.

Might be on $2^{\{\text{HW, MIDTERM, FINAL}\}}$.

8 possibilities.

Is there a smaller DFA for $\Sigma^* a \Sigma^i$? Fewer than 2^{i+1} states? No.
We may prove this later.

We now use NFA's informally.

All You Need to Know About NFA's For Now

All You Need to Know About NFA's For Now

1. From state q, and symbol σ may be ≥ 2 states to go to.

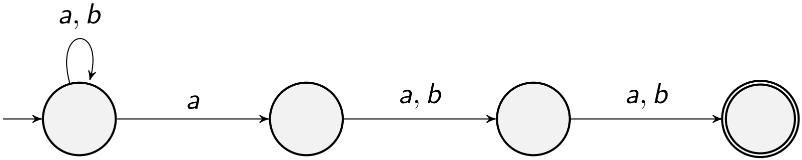
All You Need to Know About NFA's For Now

1. From state q, and symbol σ may be ≥ 2 states to go to.
2. From a state q and no symbols there may be ≥ 1 states to go to. (We use e for **empty string**.)

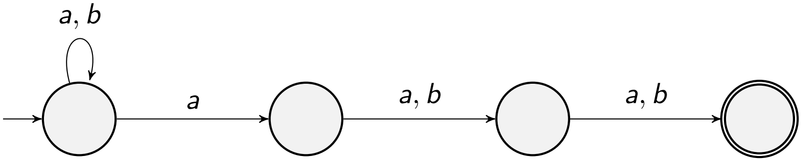
All You Need to Know About NFA's For Now

1. From state q, and symbol σ may be ≥ 2 states to go to.
2. From a state q and no symbols there may be ≥ 1 states to go to. (We use e for **empty string**.)
3. An NFA accepts a string if there is **some** way to process the string and get to a final state.

NFA for $\Sigma^* a \Sigma^2$



NFA for $\Sigma^* a \Sigma^2$



DFA had 8 states. NFA has 4 states.

NFA for $\Sigma^* a \Sigma^3$

Recall that DFA for $\Sigma^* a \Sigma^3$ used 16 states.

NFA for $\Sigma^* a \Sigma^3$

Recall that DFA for $\Sigma^* a \Sigma^3$ used 16 states.

Draw an NFA for $\Sigma^* a \Sigma^3$.

NFA for $\Sigma^* a \Sigma^3$

Recall that DFA for $\Sigma^* a \Sigma^3$ used 16 states.

Draw an NFA for $\Sigma^* a \Sigma^3$.

How many states?

NFA for $\Sigma^* a \Sigma^3$

Recall that DFA for $\Sigma^* a \Sigma^3$ used 16 states.

Draw an NFA for $\Sigma^* a \Sigma^3$.

How many states?

Make a conjecture for number of states for NFA for $\Sigma^* a \Sigma^n$.

NFA for $\Sigma^* a \Sigma^3$

Recall that DFA for $\Sigma^* a \Sigma^3$ used 16 states.

Draw an NFA for $\Sigma^* a \Sigma^3$.

How many states?

Make a conjecture for number of states for NFA for $\Sigma^* a \Sigma^n$.

Upshot Seems like NFA uses far fewer state than DFA for $\Sigma^* a \Sigma^n$.

$\{w : \#_a(w) \equiv 0 \pmod 3 \vee \#_b(w) \equiv 0 \pmod 4\}$

The DFA for this requires 12 states. Can we do this with a smaller NFA?

$\{w : \#_a(w) \equiv 0 \pmod 3 \vee \#_b(w) \equiv 0 \pmod 4\}$

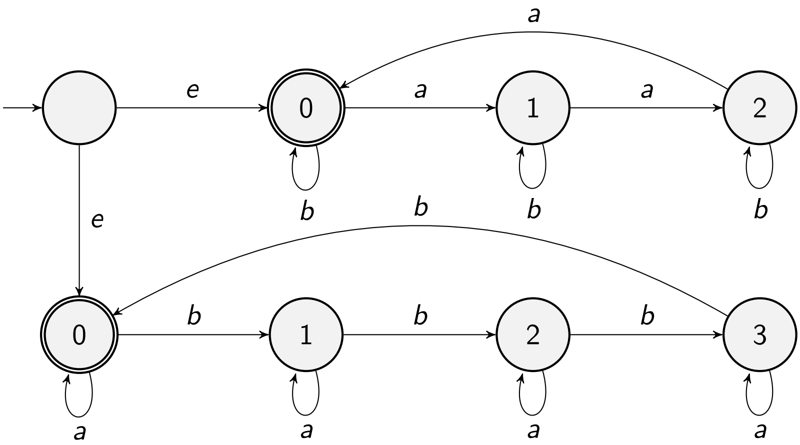
The DFA for this requires 12 states. Can we do this with a smaller NFA? **Vote**

$\{w : \#_a(w) \equiv 0 \pmod 3 \vee \#_b(w) \equiv 0 \pmod 4\}$

The DFA for this requires 12 states. Can we do this with a smaller NFA? **Vote**

YES - next slide.

$\{w : \#_a(w) \equiv 0 \pmod 3 \vee \#_b(w) \equiv 0 \pmod 4\}$



$\{w : \#_a(w) \equiv 0 \pmod 3 \wedge \#_b(w) \equiv 0 \pmod 4\}$

The DFA for this requires 12 states. Can we do this with a smaller NFA?

$\{w : \#_a(w) \equiv 0 \pmod 3 \wedge \#_b(w) \equiv 0 \pmod 4\}$

The DFA for this requires 12 states. Can we do this with a smaller NFA? **Vote**

$\{w : \#_a(w) \equiv 0 \pmod 3 \wedge \#_b(w) \equiv 0 \pmod 4\}$

The DFA for this requires 12 states. Can we do this with a smaller NFA? **Vote**

NO. Proof similar to that for DFA. May do both later.

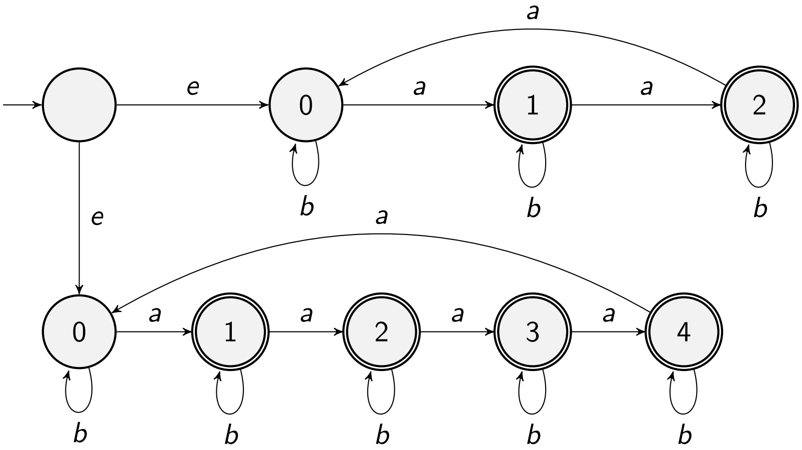
$\{a^n : n \not\equiv 0 \pmod{15}\}$

Note A DFA for this **requires** 15 states. Can a smaller NFA recognize it? **Vote**

$\{a^n : n \not\equiv 0 \pmod{15}\}$

Note A DFA for this **requires** 15 states. Can a smaller NFA recognize it? **Vote**
YES - next slide

$\{a^n : n \not\equiv 0 \pmod{15}\}$



$\{a^n : n \not\equiv 0 \pmod{15}\}$

Prove that the NFA in the last slide works.
Need

$$(n \not\equiv 0 \pmod 3 \vee n \not\equiv 0 \pmod 5) \implies n \not\equiv 0 \pmod{15}$$

Take the contrapositive

$$n \equiv 0 \pmod{15} \implies (n \equiv 0 \pmod 3 \wedge n \equiv 0 \pmod 5)$$

$\{a^n : n \equiv 0 \pmod{15}\}$

Note A DFA for this **requires** 15 states. Can a smaller NFA recognize it? **Vote**

$\{a^n : n \equiv 0 \pmod{15}\}$

Note A DFA for this **requires** 15 states. Can a smaller NFA recognize it? **Vote**

NO. Proof similar to that for DFA. May do both later, after we define NFA rigorously.

NFA's Intuitively

1. An NFA is a DFA that can guess.
2. NFAs do not really exist.
3. Good for $\cup$ since can guess which one.
4. An NFA accepts iff SOME guess accepts.
5. NFA's are **useful** as intermediary devices.

NFA Formally

Def An **NFA** is a tuple $(Q, \Sigma, \Delta, s, F)$ where:

1. Q is a finite set of **states**.
2. Σ is a finite **alphabet**.
3. $\Delta : Q \times (\Sigma \cup \{e\}) \to 2^Q$ is the *transition function*.
4. $s \in Q$ is the **start state**.
5. $F \subseteq Q$ is the set of **final states**.

NFA Formally

Def An **NFA** is a tuple $(Q, \Sigma, \Delta, s, F)$ where:

1. Q is a finite set of **states**.
2. Σ is a finite **alphabet**.
3. $\Delta : Q \times (\Sigma \cup \{e\}) \to 2^Q$ is the *transition function*.
4. $s \in Q$ is the **start state**.
5. $F \subseteq Q$ is the set of **final states**.

Def If M is an NFA and $x \in \Sigma^*$ then $\boldsymbol{M(x)}$ **accepts** if when you run M on x **some sequence of guesses** end up in a **final state**.

NFA Formally

Def An **NFA** is a tuple $(Q, \Sigma, \Delta, s, F)$ where:

1. Q is a finite set of **states**.
2. Σ is a finite **alphabet**.
3. $\Delta : Q \times (\Sigma \cup \{e\}) \rightarrow 2^Q$ is the *transition function*.
4. $s \in Q$ is the **start state**.
5. $F \subseteq Q$ is the set of **final states**.

Def If M is an NFA and $x \in \Sigma^*$ then $\boldsymbol{M(x)}$ **accepts** if when you run M on x **some sequence of guesses** end up in a **final state**.

Note When you run $M(x)$ and choose a path one of three things can happen: (1) ends in a final state, (2) ends in a non-final state, (3) cannot process.

NFA Formally

Def An **NFA** is a tuple $(Q, \Sigma, \Delta, s, F)$ where:

1. Q is a finite set of **states**.
2. Σ is a finite **alphabet**.
3. $\Delta : Q \times (\Sigma \cup \{e\}) \rightarrow 2^Q$ is the *transition function*.
4. $s \in Q$ is the **start state**.
5. $F \subseteq Q$ is the set of **final states**.

Def If M is an NFA and $x \in \Sigma^*$ then $\boldsymbol{M(x)}$ **accepts** if when you run M on x **some sequence of guesses** end up in a **final state**.
Note When you run $M(x)$ and choose a path one of three things can happen: (1) ends in a final state, (2) ends in a non-final state, (3) cannot process.
Def If M is an NFA then $L(M) = \{x : M(x) \text{ accepts } \}$.

Three Way to Think About NFAs

Three Way to Think About NFAs

- **Computational (with parallelism):** Fork new computational threads whenever there is a choice. Accept if any thread accepts.

Three Way to Think About NFAs

- **Computational (with parallelism):** Fork new computational threads whenever there is a choice. Accept if any thread accepts.
- **Mathematical:** Create tree with branches whenever there is a choice. Accept if any leaf accepts.

Three Way to Think About NFAs

- **Computational (with parallelism):** Fork new computational threads whenever there is a choice. Accept if any thread accepts.
- **Mathematical:** Create tree with branches whenever there is a choice. Accept if any leaf accepts.
- **Magic:** Guess at each nondeterministic step which way to go. Machine always makes right guess if there is one.

Is Every NFA-lang a DFA-lang?

Is Every NFA-lang a DFA-lang?

1. We have seen several langs where the NFA is smaller than the DFA.

Is Every NFA-lang a DFA-lang?

1. We have seen several langs where the NFA is smaller than the DFA.
2. We have NOT seen any langs that an NFA can accept but a DFA cannot accept.

Is Every NFA-lang a DFA-lang?

1. We have seen several langs where the NFA is smaller than the DFA.
2. We have NOT seen any langs that an NFA can accept but a DFA cannot accept.

SO, is every NFA-lang also a DFA-lang?

Is Every NFA-lang a DFA-lang?

1. We have seen several langs where the NFA is smaller than the DFA.
2. We have NOT seen any langs that an NFA can accept but a DFA cannot accept.

SO, is every NFA-lang also a DFA-lang? **Vote**.

Is Every NFA-lang a DFA-lang?

1. We have seen several langs where the NFA is smaller than the DFA.
2. We have NOT seen any langs that an NFA can accept but a DFA cannot accept.

SO, is every NFA-lang also a DFA-lang? **Vote**. Yes.

Every NFA-lang a DFA-lang!

Thm If L is accepted by an NFA then L is accepted by a DFA.

Every NFA-lang a DFA-lang!

Thm If L is accepted by an NFA then L is accepted by a DFA.
Pf L is accepted by NFA $(Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times (\Sigma \cup \{e\}) \to 2^Q$.

Every NFA-lang a DFA-lang!

Thm If L is accepted by an NFA then L is accepted by a DFA.
Pf L is accepted by NFA $(Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times (\Sigma \cup \{e\}) \rightarrow 2^Q$.
First we get rid of the e-transitions.

Every NFA-lang a DFA-lang!

Thm If L is accepted by an NFA then L is accepted by a DFA.
Pf L is accepted by NFA $(Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times (\Sigma \cup \{e\}) \rightarrow 2^Q$.
First we get rid of the e-transitions.
Notation $\Delta(q, e^i \sigma e^j)$ means that we take state q, feed in e i times, then feed in σ, then feed in e j times. Do all possible transitions so this will be a set of states.

Every NFA-lang a DFA-lang!

Thm If L is accepted by an NFA then L is accepted by a DFA.
Pf L is accepted by NFA $(Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times (\Sigma \cup \{e\}) \rightarrow 2^Q$.
First we get rid of the e-transitions.
Notation $\Delta(q, e^i \sigma e^j)$ means that we take state q, feed in e i times, then feed in σ, then feed in e j times. Do all possible transitions so this will be a set of states.

$$\Delta_1(q, \sigma) = \bigcup_{0 \le i,j \le n} \Delta(q, e^i \sigma e^j).$$

Every NFA-lang a DFA-lang!

Thm If L is accepted by an NFA then L is accepted by a DFA.
Pf L is accepted by NFA $(Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times (\Sigma \cup \{e\}) \rightarrow 2^Q$.
First we get rid of the e-transitions.
Notation $\Delta(q, e^i \sigma e^j)$ means that we take state q, feed in e i times, then feed in σ, then feed in e j times. Do all possible transitions so this will be a set of states.

$$\Delta_1(q, \sigma) = \bigcup_{0 \le i,j \le n} \Delta(q, e^i \sigma e^j).$$

NFA $(Q, \Sigma, \Delta_1, s, F)$ accepts same lang as $(Q, \Sigma, \Delta, s, F)$.

Every NFA-lang a DFA-lang!

Thm If L is accepted by an NFA then L is accepted by a DFA.
Pf L is accepted by NFA $(Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times (\Sigma \cup \{e\}) \rightarrow 2^Q$.
First we get rid of the e-transitions.
Notation $\Delta(q, e^i \sigma e^j)$ means that we take state q, feed in e i times, then feed in σ, then feed in e j times. Do all possible transitions so this will be a set of states.

$$\Delta_1(q, \sigma) = \bigcup_{0 \le i,j \le n} \Delta(q, e^i \sigma e^j).$$

NFA $(Q, \Sigma, \Delta_1, s, F)$ accepts same lang as $(Q, \Sigma, \Delta, s, F)$.
We will work with an NFA that has NO e-transitions.

Every NFA-lang a DFA-lang!

Thm If L is accepted by an NFA then L is accepted by a DFA.
Pf L is accepted by NFA $(Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times (\Sigma \cup \{e\}) \rightarrow 2^Q$.
First we get rid of the e-transitions.
Notation $\Delta(q, e^i \sigma e^j)$ means that we take state q, feed in e i times, then feed in σ, then feed in e j times. Do all possible transitions so this will be a set of states.

$$\Delta_1(q, \sigma) = \bigcup_{0 \le i,j \le n} \Delta(q, e^i \sigma e^j).$$

NFA $(Q, \Sigma, \Delta_1, s, F)$ accepts same lang as $(Q, \Sigma, \Delta, s, F)$.
We will work with an NFA that has NO e-transitions.
We are nowhere near done. Next slide.

Every NFA-lang a DFA-lang! (Cont)

Thm If L is accepted by an NFA with n states and no e-transitions then L is accepted by a DFA with $\leq 2^n$ states.

Every NFA-lang a DFA-lang! (Cont)

Thm If L is accepted by an NFA with n states and no e-transitions then L is accepted by a DFA with $\leq 2^n$ states.
Pf L is accepted by NFA $M = (Q, \Sigma, \Delta, s, F)$ where $\Delta : Q \times \Sigma \rightarrow 2^Q$.

Every NFA-lang a DFA-lang! (Cont)

Thm If L is accepted by an NFA with n states and no e-transitions then L is accepted by a DFA with $\leq 2^n$ states.
Pf L is accepted by NFA $M = (Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times \Sigma \rightarrow 2^Q$.
We define a DFA that recognizes the same language as M.

Every NFA-lang a DFA-lang! (Cont)

Thm If L is accepted by an NFA with n states and no e-transitions then L is accepted by a DFA with $\leq 2^n$ states.
Pf L is accepted by NFA $M = (Q, \Sigma, \Delta, s, F)$ where $\Delta : Q \times \Sigma \rightarrow 2^Q$.
We define a DFA that recognizes the same language as M.
Key The DFA will keep track of the **set** of states that the NFA could have been in.

Every NFA-lang a DFA-lang! (Cont)

Thm If L is accepted by an NFA with n states and no e-transitions then L is accepted by a DFA with $\le 2^n$ states.

Pf L is accepted by NFA $M = (Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times \Sigma \to 2^Q$.

We define a DFA that recognizes the same language as M.

Key The DFA will keep track of the **set** of states that the NFA could have been in.

DFA $(2^Q, \Sigma, \delta, \{s\}, F')$. Need to define δ and F'.

Every NFA-lang a DFA-lang! (Cont)

Thm If L is accepted by an NFA with n states and no e-transitions then L is accepted by a DFA with $\leq 2^n$ states.

Pf L is accepted by NFA $M = (Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times \Sigma \to 2^Q$.

We define a DFA that recognizes the same language as M.

Key The DFA will keep track of the **set** of states that the NFA could have been in.

DFA $(2^Q, \Sigma, \delta, \{s\}, F')$. Need to define δ and F'.
$\delta : 2^Q \times \Sigma \to 2^Q$.

Every NFA-lang a DFA-lang! (Cont)

Thm If L is accepted by an NFA with n states and no e-transitions then L is accepted by a DFA with $\le 2^n$ states.
Pf L is accepted by NFA $M = (Q, \Sigma, \Delta, s, F)$ where $\Delta : Q \times \Sigma \rightarrow 2^Q$.
We define a DFA that recognizes the same language as M.
Key The DFA will keep track of the **set** of states that the NFA could have been in.
DFA $(2^Q, \Sigma, \delta, \{s\}, F')$. Need to define δ and F'.
$\delta : 2^Q \times \Sigma \rightarrow 2^Q$.

$$\delta(A, \sigma) = \bigcup_{q \in A} \Delta(q, \sigma).$$

Every NFA-lang a DFA-lang! (Cont)

Thm If L is accepted by an NFA with n states and no e-transitions then L is accepted by a DFA with $\le 2^n$ states.
Pf L is accepted by NFA $M = (Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times \Sigma \to 2^Q$.
We define a DFA that recognizes the same language as M.
Key The DFA will keep track of the **set** of states that the NFA could have been in.
DFA $(2^Q, \Sigma, \delta, \{s\}, F')$. Need to define δ and F'.
$\delta : 2^Q \times \Sigma \to 2^Q$.

$$\delta(A, \sigma) = \bigcup_{q \in A} \Delta(q, \sigma).$$

$$F' = \{A : A \cap F \neq \emptyset\}.$$

Every NFA-lang a DFA-lang! (Cont)

Thm If L is accepted by an NFA with n states and no e-transitions then L is accepted by a DFA with $\le 2^n$ states.
Pf L is accepted by NFA $M = (Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times \Sigma \to 2^Q$.
We define a DFA that recognizes the same language as M.
Key The DFA will keep track of the **set** of states that the NFA could have been in.
DFA $(2^Q, \Sigma, \delta, \{s\}, F')$. Need to define δ and F'.
$\delta : 2^Q \times \Sigma \to 2^Q$.

$$\delta(A, \sigma) = \bigcup_{q \in A} \Delta(q, \sigma).$$

$$F' = \{A : A \cap F \neq \emptyset\}.$$

If NFA accepts on some path then in the DFA you will be in a state which is a set-of-states, which includes a final state from the NFA.

Every NFA-lang a DFA-lang! (Cont)

Thm If L is accepted by an NFA with n states and no e-transitions then L is accepted by a DFA with $\leq 2^n$ states.

Pf L is accepted by NFA $M = (Q, \Sigma, \Delta, s, F)$ where
$\Delta : Q \times \Sigma \to 2^Q$.

We define a DFA that recognizes the same language as M.

Key The DFA will keep track of the **set** of states that the NFA could have been in.

DFA $(2^Q, \Sigma, \delta, \{s\}, F')$. Need to define δ and F'.
$\delta : 2^Q \times \Sigma \to 2^Q$.

$$\delta(A, \sigma) = \bigcup_{q \in A} \Delta(q, \sigma).$$

$$F' = \{A : A \cap F \neq \emptyset\}.$$

If NFA accepts on some path then in the DFA you will be in a state which is a set-of-states, which includes a final state from the NFA. If the DFA accepts then there was some way for the NFA to accept.

BILL, STOP RECORDING LECTURE!!!!

BILL STOP RECORDING LECTURE!!!