BILL, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!
Nondeterministic Finite Automata (NFA)
An Interesting Example of a DFA

In breakout rooms do the following and keep track of how many states.

\[ \Sigma^* a \]
\[ \Sigma^* a\Sigma \]
\[ \Sigma^* a\Sigma^2 \]
\[ \Sigma^* a \Sigma^2 \]

The number of states is 8.

The number of states is 8.

More generally:
$\Sigma^* a\Sigma^i$ can be done with $2^{i+1}$ states.
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.
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\{HW, MIDTERM, FINAL\}.
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 \( 2\{HW, \text{ MIDTERM, FINAL}\} \).

8 possibilities.
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 \( 2\{HW, MIDTERM, FINAL\} \).
8 possibilities.
Is there a smaller DFA for \( \Sigma^* a \Sigma^i \)? Fewer than \( 2^{i+1} \) states?
The number of states is 8.
More generally:
Σ^* aΣ^i can be done with 2^{i+1} states.
Prove for Σ^* aΣ^3, with a table.
Might be on 2{HW, MIDTERM, FINAL}.
8 possibilities.
Is there a smaller DFA for Σ^* aΣ^i? Fewer than 2^{i+1} states? No.
We may prove this later.
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\{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.
NFA for $\Sigma^* a \Sigma^2$
\[\{w : \#_a \equiv 0 \pmod{3} \lor w : \#_b \equiv 0 \pmod{4}\}\]

The DFA for this requires 12 states. Can we do this with a smaller NFA?
\{ w : \#_a \equiv 0 \pmod{3} \lor w : \#_b \equiv 0 \pmod{4} \}\}

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

YES - next slide.
\{ \begin{align*} w : \#_a &\equiv 0 \pmod{3} \lor \ w : \#_b &\equiv 0 \pmod{4} \end{align*} \}
\{ w : \#_a \equiv 0 \pmod{3} \land w : \#_b \equiv 0 \pmod{4} \} 

The DFA for this requires 12 states. Can we do this with a smaller NFA?
\{ w : \#_a \equiv 0 \pmod{3} \land w : \#_b \equiv 0 \pmod{4} \} \\

The DFA for this requires 12 states. Can we do this with a smaller NFA? Vote!
\( \{ w : \#_a \equiv 0 \pmod{3} \land w : \#_b \equiv 0 \pmod{4} \} \)

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

**NO.** Proof similar to that for DFA. Will come back to this after we define NFA rigorously.
\{ w : \#_a \equiv 0 \ (\text{mod} \ 3) \land w : \#_b \equiv 0 \ (\text{mod} \ 4) \}\}

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

**NO.** Proof similar to that for DFA. Will come back to this after we define NFA rigorously.

Or might be on HW-MID-FINAL.
\{a^n \mid 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}\}
Prove that the NFA in the last slide works.

Need

\[(n \not\equiv 0 \pmod{3} \lor n \not\equiv 0 \pmod{5}) \implies n \not\equiv 0 \pmod{15}\]

Take the contrapositive

\[n \equiv 0 \pmod{15} \implies (n \equiv 0 \pmod{3} \land n \equiv 0 \pmod{5})\]
\{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. Will come back to this after we define NFA rigorously.
\{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. Will come back to this after we define NFA rigorously.

Or might be on HW-MID-FINAL.
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.
Def An NFA is a tuple \((Q, \Sigma, \Delta, s, F)\) where:

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

**Def** If \(M\) is an NFA and \(x \in \Sigma^*\) then \(M(x)\) 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. \(\Sigma\) is a finite **alphabet**.
3. \(\Delta : Q \times (\Sigma \cup \{e\}) \to 2^Q\) is the **transition function**.
4. \(s \in S\) is the **start state**.
5. \(F \in Q\) is the set of **final states**.

**Def** If \(M\) is an NFA and \(x \in \Sigma^*\) then \(M(x)\) **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. \(\Sigma\) is a finite **alphabet**.
3. \(\Delta : Q \times (\Sigma \cup \{e\}) \rightarrow 2^Q\) is the **transition function**.
4. \(s \in S\) is the **start state**.
5. \(F \in Q\) is the set of **final states**.

**Def** If \(M\) is an NFA and \(x \in \Sigma^*\) then **\(M(x)\ accepts\)** 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 **\(L(M) = \{x : M(x)\ accepts\}\)**.
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 do but a DFA cannot do.

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

Vote: Yes.
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 do but a DFA cannot do.
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 do but a DFA cannot do.

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 do but a DFA cannot do.

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 do but a DFA cannot do.

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.

**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 $\sigma$, then feed in $e^j$ times. Do all possible transitions so this will be a set of states.

$\Delta_1(q, \sigma) = \bigcup_{0 \leq i, j \leq n} \Delta(q, e^i \sigma e^j)$.
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 $\sigma$, then feed in $e_j$ times. Do all possible transitions so this will be a set of states.

$\Delta_1(q, \sigma) = \bigcup_{0 \leq i, j \leq n} \Delta(q, e_i \sigma e_j)$. 

We will work with an NFA that has NO $e$-transitions.

We are nowhere near done. Next slide.
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 $\sigma$, then feed in $e^j$ times. Do all possible transitions so this will be a set of states.

**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 $\sigma$, then feed in $e$ $j$ times. Do all possible transitions so this will be a set of states.

$$\Delta_1(q, \sigma) = \bigcup_{0 \leq i, j \leq n} \Delta(q, e^i\sigma e^j).$$
**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 \( \sigma \), then feed in \( e \) \( j \) times. Do all possible transitions so this will be a set of states.

\[
\Delta_1(q, \sigma) = \bigcup_{0 \leq i, j \leq n} \Delta(q, e^i\sigma e^j).
\]

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 \(\Delta: Q \times (\Sigma \cup \{e\}) \to 2^Q\).

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 \(\sigma\), then feed in \(e\) \(j\) times. Do all possible transitions so this will be a set of states.

\[
\Delta_1(q, \sigma) = \bigcup_{0 \leq i, j \leq n} \Delta(q, e^i \sigma e^j).
\]

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

$\Delta : Q \times (\Sigma \cup \{e\}) \to 2^Q$.

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 $\sigma$, then feed in $e^j$ times. Do all possible transitions so this will be a set of states.

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

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 on $n$ states then $L$ is accepted by a DFA on $\leq 2^n$ 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 on $n$ states then $L$ is accepted by a DFA on $\leq 2^n$ 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 $M$. 

### DFA $(2^Q, \Sigma, \delta, \{s\}, F')$.

- **$\delta$** is defined as $\delta(A, \sigma) = \bigcup_{q \in A} \Delta(q, \sigma)$.
- **$F'$** is defined as $F' = \{A : A \cap F \neq \emptyset\}$.

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

**Thm** If \( L \) is accepted by an NFA on \( n \) states then \( L \) is accepted by a DFA on \( \leq 2^n \) 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 on $n$ states then $L$ is accepted by a DFA on $\leq 2^n$ 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 $\delta$ and $F'$. 
**Every NFA-lang a DFA-lang! (Cont)**

**Thm** If $L$ is accepted by an NFA on $n$ states then $L$ is accepted by a DFA on $\leq 2^n$ 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.

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

$\delta : 2^Q \times \Sigma \rightarrow 2^Q$. 
Every NFA-lang a DFA-lang! (Cont)

**Thm** If $L$ is accepted by an NFA on $n$ states then $L$ is accepted by a DFA on $\leq 2^n$ 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 $\delta$ and $F'$.

$\delta : 2^Q \times \Sigma \to 2^Q$.

\[
\delta(A, \sigma) = \bigcup_{q \in A} \Delta(q, \sigma).
\]
**Thm** If $L$ is accepted by an NFA on $n$ states then $L$ is accepted by a DFA on $\leq 2^n$ 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.

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

$\delta : 2^Q \times \Sigma \rightarrow 2^Q$. 

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

$F' = \{A : A \cap F \neq \emptyset\}.$
Every NFA-lang a DFA-lang! (Cont)

**Thm** If $L$ is accepted by an NFA on $n$ states then $L$ is accepted by a DFA on $\leq 2^n$ 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.

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

$\delta : 2^Q \times \Sigma \rightarrow 2^Q$.

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

$$F' = \{A : A \cap F \neq \emptyset\}.$$ 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 on \( n \) states then \( L \) is accepted by a DFA on \( \leq 2^n \) 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.

**DFA** \((2^Q, \Sigma, \delta, \{s\}, F')\). Need to define \( \delta \) and \( F' \).
\[ \delta : 2^Q \times \Sigma \rightarrow 2^Q. \]

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

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

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 say for the NFA to accept.
BILL, STOP RECORDING LECTURE!!!!

BILL STOP RECORDING LECTURE!!!