Nondeterminism and Subset Construction

Nondeterministic Finite Automaton

- Nondeterminism: More than one state transitions may be available from any current state. The machine can choose the next transition from options.

Nondeterministic Finite Automaton

- Nondeterminism: More than one state transitions may be available from any current state. The machine can choose the next transition from options.
- A transition function is no longer required, should be a map, since there are many possibilities.

Nondeterministic Finite Automaton

- Nondeterminism: More than one state transitions may be available from any current state. The machine can choose the next transition from options.
- A transition function is no longer required, should be a map, since there are many possibilities.
- We could guess where to start from; so, many start states are possible. No longer a single start state.

Nondeterministic Finite Automaton

- The idea is that the automaton guesses a path in it and verifies whether that guess leads to a final state. The other paths may lead to non-final states or may end even before you finish reading the input (terminated).

Nondeterministic Finite Automaton

- The idea is that the automaton guesses a path in it and verifies whether that guess leads to a final state. The other paths may lead to non-final states or may end even before you finish reading the input (terminated).
- Note 1: You are allowed to put more than one arrows with the same label coming out of a state.

Nondeterministic Finite Automaton

- The idea is that the automaton guesses a path in it and verifies whether that guess leads to a final state. The other paths may lead to non-final states or may end even before you finish reading the input (terminated).
- Note 1: You are allowed to put more than one arrows with the same label coming out of a state.
- Note 2: You are allowed to choose a set of start states.

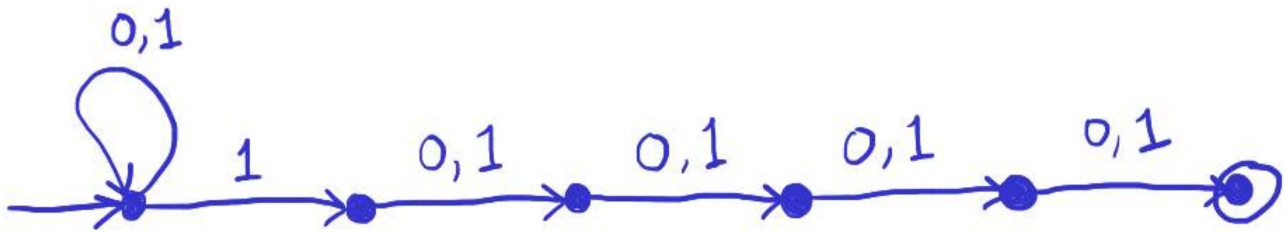
Nondeterministic Finite Automaton

- The idea is that the automaton guesses a path in it and verifies whether that guess leads to a final state. The other paths may lead to non-final states or may end even before you finish reading the input (terminated).
- Note 1: You are allowed to put more than one arrows with the same label coming out of a state.
- Note 2: You are allowed to choose a set of start states.
- Note 3: Since we are relaxing the transition function to a transition map and allowing termination of a guess, we do not need to define transitions from a state for each possible input alphabet – there may be no option for a transition from a certain state on reading a certain alphabet!

Nondeterministic Finite Automaton (NFA): Example

$A = \{ x \in \{0,1\}^* \mid \text{the fifth symbol from the right is } 1 \}$

N s.t N accepts A



Power of an NFA

- Much more compact that a corresponding DFA.

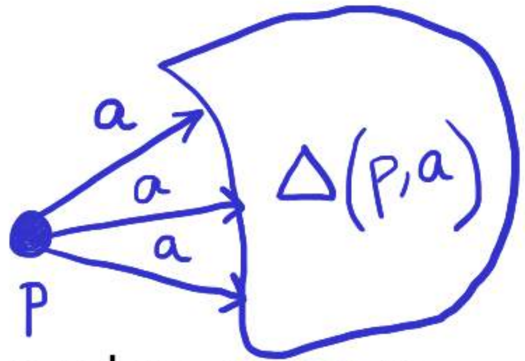
Power of an NFA

- Much more compact that a corresponding DFA.
- But does an NFA really have more computation power than a DFA?

Power of an NFA

- Much more compact that a corresponding DFA.
- But does an NFA really have more computation power than a DFA?
- Can there be a language that is not a regular set (no DFA) but has an NFA?

Formal definition of NFA



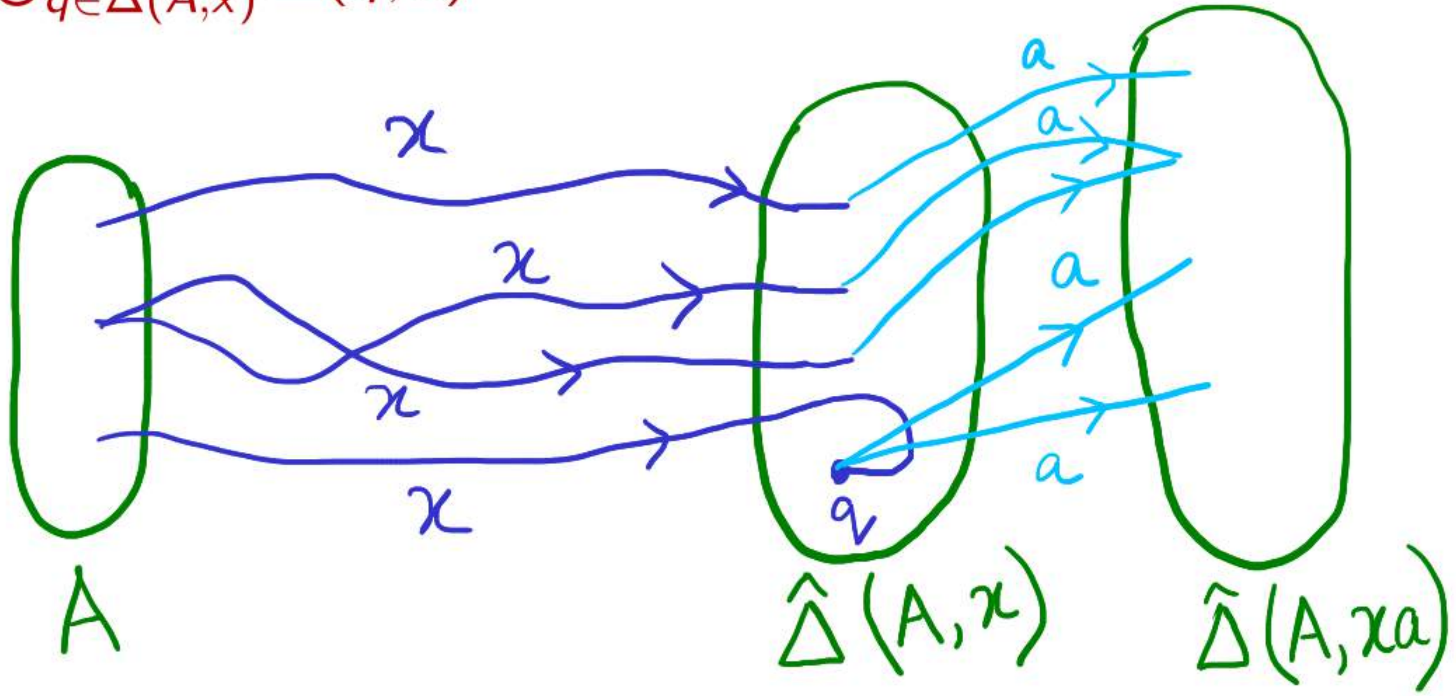
- $N = (Q, \Sigma, \Delta, S, F)$
- Q – states
- Σ – alphabet
- $\Delta : Q \times \Sigma \to 2^Q$, all states that N is allowed to move to from state p on reading input a.
- S – start states
- F – final states.
- Again, finite encoding of an NFA possible.

Multistep extension of Delta

- $\hat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$

Multistep extension of Delta

- $\hat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$
- Inductive definition:
  $\hat{\Delta}(A, \epsilon) = A$
  $\hat{\Delta}(A, xa) = \bigcup_{q \in \hat{\Delta}(A,x)} \Delta(q, a)$



Multistep extension of Delta

- $\hat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$
- Inductive definition:
  $\hat{\Delta}(A, \epsilon) = A$
  $\hat{\Delta}(A, xa) = \bigcup_{q \in \hat{\Delta}(A,x)} \Delta(q, a)$
- Image set of states = set of all states reachable from some state in A on reading input xa.

Multistep extension of Delta

- $\hat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$
- Inductive definition:
  $\hat{\Delta}(A, \epsilon) = A$
  $\hat{\Delta}(A, xa) = \bigcup_{q \in \hat{\Delta}(A,x)} \Delta(q, a)$
- Image set of states = set of all states reachable from some state in A on reading input xa.
- State $r \in \hat{\Delta}(A, xa)$ if there exists a $q \in \hat{\Delta}(A, x)$ such that $r \in \Delta(q, a)$.

Multistep extension of Delta

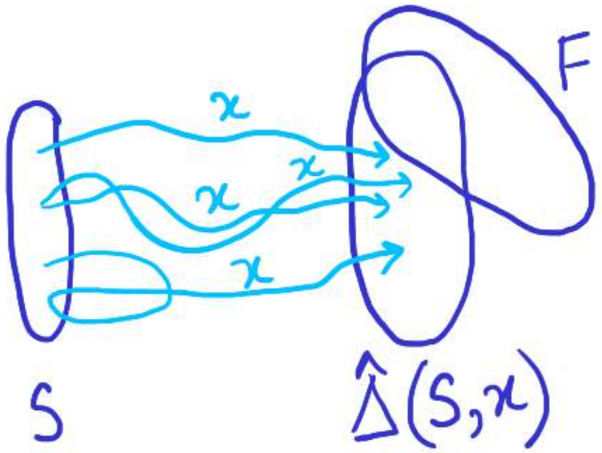
- What happens on length 1 strings?

Multistep extension of Delta

- What happens on length 1 strings?
- $\hat{\Delta}(A, a) = \hat{\Delta}(A, \epsilon.a) = \bigcup_{q \in \hat{\Delta}(A,\epsilon)} \Delta(q, a) = \bigcup_{q \in A} \Delta(q, a)$

Multistep extension of Delta

- What happens on length 1 strings?
- $\hat{\Delta}(A, a) = \hat{\Delta}(A, \epsilon.a) = \bigcup_{q \in \hat{\Delta}(A,\epsilon)} \Delta(q, a) = \bigcup_{q \in A} \Delta(q, a)$
- N accepts x if $\hat{\Delta}(S, x) \cap F$ is nonempty.



Multistep extension of Delta

- What happens on length 1 strings?
- $\hat{\Delta}(A, a) = \hat{\Delta}(A, \epsilon.a) = \bigcup_{q \in \hat{\Delta}(A,\epsilon)} \Delta(q, a) = \bigcup_{q \in A} \Delta(q, a)$
- N accepts x if $\hat{\Delta}(S, x) \cap F$ is nonempty.
- $L(N) = \{x | N \text{ accepts } x\}$

- **Lemma**: For any $x, y \in \Sigma^*$, and a subset $A \subseteq Q$, $\hat{\Delta}(A, xy) = \hat{\Delta}(\hat{\Delta}(A, x), y)$.

- **Lemma**: For any $x, y \in \Sigma^*$, and a subset $A \subseteq Q$, $\hat{\Delta}(A, xy) = \hat{\Delta}(\hat{\Delta}(A, x), y)$.
- Proof: Induction on $|y|$.

- **Lemma**: For any $x, y \in \Sigma^*$, and a subset $A \subseteq Q$, $\hat{\Delta}(A, xy) = \hat{\Delta}(\hat{\Delta}(A, x), y)$.
- Proof: Induction on $|y|$.
- Base: $y = \epsilon$
  $\hat{\Delta}(A, x.\epsilon)$
  $= \hat{\Delta}(A, x)$
  $= \hat{\Delta}(\hat{\Delta}(A, x), \epsilon)$ [By definition].

- Induction: Suppose true for all x and y; take ya.

- Induction: Suppose true for all x and y; take ya.
- $\hat{\Delta}(A, xya) = \bigcup_{q \in \hat{\Delta}(A,xy)} \Delta(q, a)$
  $= \bigcup_{q \in \hat{\Delta}(\hat{\Delta}(A,x),y)} \Delta(q, a)$ [I.H]
  $= \hat{\Delta}(\hat{\Delta}(A, x), ya)$ [By LHS of inductive definition]

- **Lemma**: $\hat{\Delta}(A, x) = \bigcup_{p \in A} \hat{\Delta}(\{p\}, x)$.

- **Lemma**: $\hat{\Delta}(A, x) = \bigcup_{p \in A} \hat{\Delta}(\{p\}, x)$.
- Proof: Induction on $|x|$.

Do NFAs have more power than DFAs?

- A DFA can be thought of as an NFA: $(Q, \Sigma, \Delta, \{s\}, F)$ such that $\Delta(p, a) = \{\delta(p, a)\}$. If a set is accepted by a DFA then it is also accepted by an NFA.

Do NFAs have more power than DFAs?

- A DFA can be thought of as an NFA: $(Q, \Sigma, \Delta, \{s\}, F)$ such that $\Delta(p, a) = \{\delta(p, a)\}$. If a set is accepted by a DFA then it is also accepted by an NFA.
- Suppose $N = (Q_N, \Sigma, \Delta_N, S_N, F_N)$ is an NFA and $L(N)$ is the set accepted by N.

Do NFAs have more power than DFAs?

- DFA construction: $M = (2^{Q_N}, \Sigma, \delta_M, s_M, F_M)$.

Do NFAs have more power than DFAs?

- DFA construction: $M = (2^{Q_N}, \Sigma, \delta_M, s_M, F_M)$.
- s_M = state corresponding to the subset S_N,

Do NFAs have more power than DFAs?

- DFA construction: $M = (2^{Q_N}, \Sigma, \delta_M, s_M, F_M)$.
- s_M = state corresponding to the subset S_N,
- F_M = states corresponding to subsets that have nonempty intersection with F_N

Do NFAs have more power than DFAs?

- DFA construction: $M = (2^{Q_N}, \Sigma, \delta_M, s_M, F_M)$.
- s_M = state corresponding to the subset S_N,
- F_M = states corresponding to subsets that have nonempty intersection with F_N
- $\delta_M(A, a) = \hat{\Delta}_N(A, a)$

- **Lemma**: For any subset $A \in 2^{Q_N}$, and string x, $\hat{\delta}_M(A, x) = \hat{\Delta}_N(A, x)$

- **Lemma**: For any subset $A \in 2^{Q_N}$, and string x, $\hat{\delta}_M(A, x) = \hat{\Delta}_N(A, x)$
- Proof: Induction on $|x|$.

- **Lemma**: For any subset $A \in 2^{Q_N}$, and string x, $\hat{\delta}_M(A, x) = \hat{\Delta}_N(A, x)$
- Proof: Induction on $|x|$.
- Base: $x = \epsilon$.
  $\hat{\delta}_M(A, \epsilon) = A = \hat{\Delta}_N(A, \epsilon)$ [Definition]

- Induction: True for all x, now consider xa.

- Induction: True for all x, now consider xa.
- $\hat{\delta}_M(A, xa)$
  $= \delta_M(\hat{\delta}_M(A, x), a)$
  $= \delta_M(\hat{\Delta}_N(A, x), a)$ [I.H]
  $= \hat{\Delta}_N(\hat{\Delta}_N(A, x), a)$ [Definition of δ_M]
  $= \hat{\Delta}_N(A, xa)$ (Earlier Lemma)

- **Theorem**: The same set of strings is accepted by M and N.

- **Theorem**: The same set of strings is accepted by M and N.
- Proof: $x \in L(M) \implies \hat{\delta}_M(s_M, x) \in F_M$
  $\iff \hat{\delta}_M(s_M, x)$ has nonempty intersection with F_N
  $\iff \hat{\Delta}_N(S_N, x) \cap F_N$ not empty [Previous Lemma]
  $\iff x \in L(N)$.

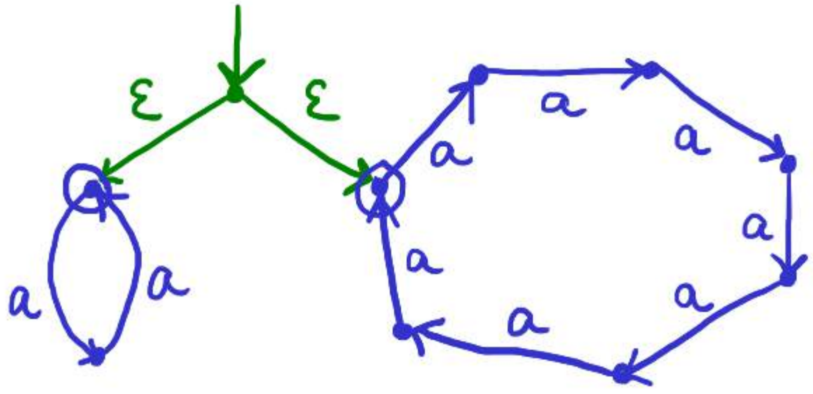
- **Theorem**: The same set of strings is accepted by M and N.
- Proof: $x \in L(M) \implies \hat{\delta}_M(s_M, x) \in F_M$
  $\iff \hat{\delta}_M(s_M, x)$ has nonempty intersection with F_N
  $\iff \hat{\Delta}_N(S_N, x) \cap F_N$ not empty [Previous Lemma]
  $\iff x \in L(N)$.
- So DFA and NFA have the same power. **NFAs accept regular sets.**

ϵ-Transitions

- A transition with label ϵ. It means that the automaton is making such a transition without reading an input symbol.

ϵ-Transitions

- A transition with label ϵ. It means that the automaton is making such a transition without reading an input symbol.
- Eg. $\{x \in \{a\}^* \mid |x| \text{ is divisible by 2 or 7}\}$

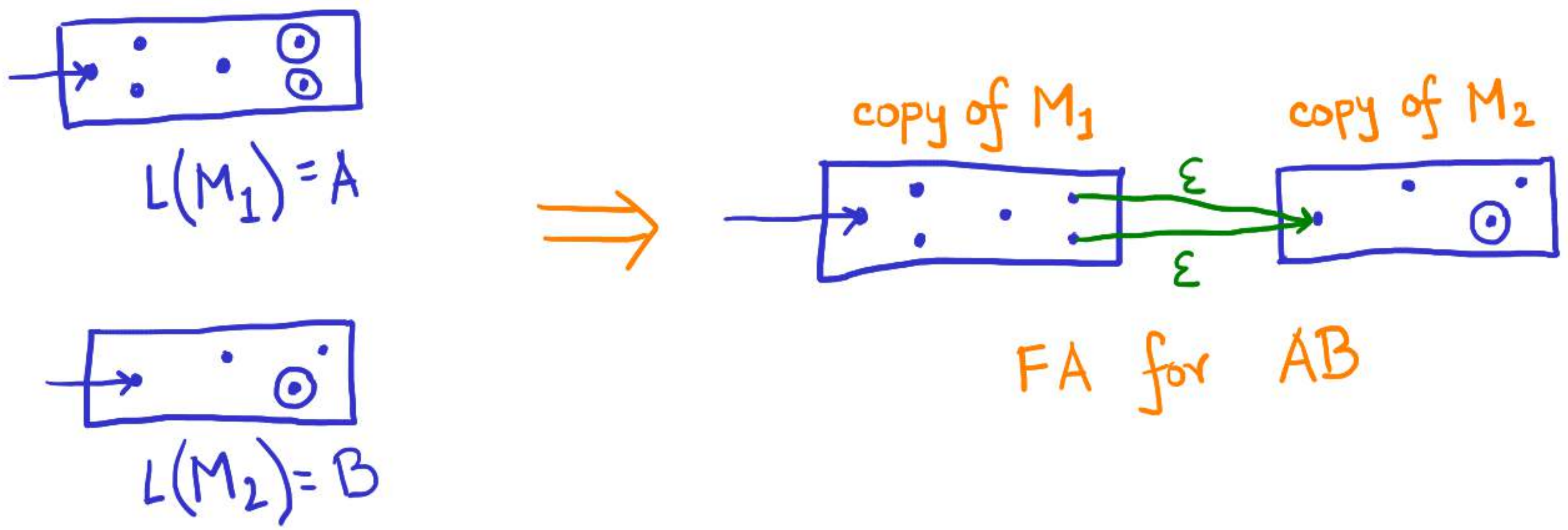


ϵ-Transitions

- A transition with label ϵ. It means that the automaton is making such a transition without reading an input symbol.
- Eg. $\{x \in \{a\}^* \mid |x| \text{ is divisible by 2 or 7}\}$
- They do not add any power. (Prove this.)

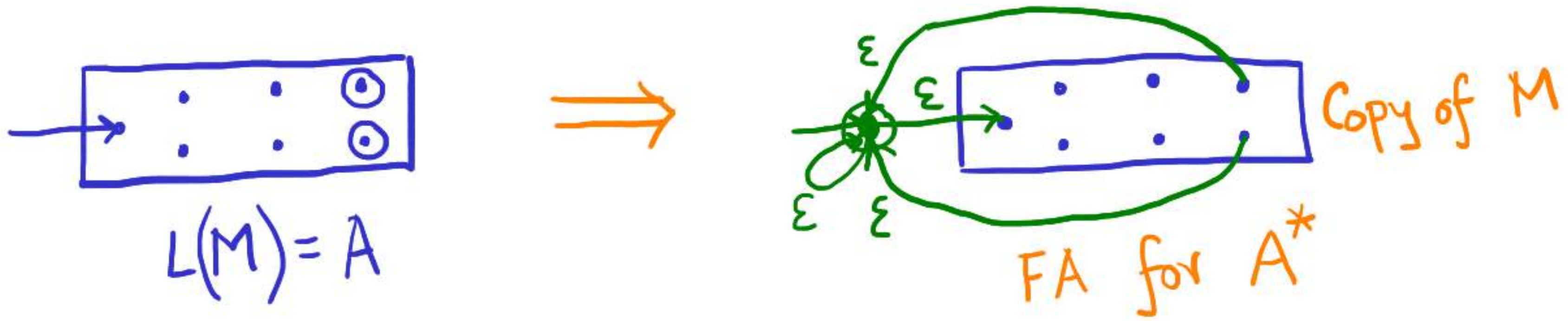
Revisiting Closure Properties of Regular Sets

- Construction of DFA/NFA for AB when A and B are regular: Construct an NFA with ϵ-transition.
  DFAs M_1 for A and M_2 for B.
  ϵ-transition between end states of A and start state of B.



Revisiting Closure Properties of Regular Sets

- Construction of DFA/NFA for AB when A and B are regular: Construct an NFA with ϵ-transition.
  DFAs M_1 for A and M_2 for B.
  ϵ-transition between end states of A and start state of B.
- If A is regular then so is A^*: Create new start state;
  Add ϵ-transitions from new start state to old start state, and from old final states to new start state
  Make new start state new final state as well.



$L(M) = A$
$\Longrightarrow$
ε
ε
ε
ε
Copy of M
FA for A^*