CS 611: Theory of Computation

Hongmin Li

Department of Computer Science
California State University, East Bay

Expressive Power of NFAs and DFAs

- Is there a language that is recognized by a DFA but not by any NFAs?

Expressive Power of NFAs and DFAs

- Is there a language that is recognized by a DFA but not by any NFAs? No!

Expressive Power of NFAs and DFAs

- Is there a language that is recognized by a DFA but not by any NFAs? No!
- Is there a language that is recognized by an NFA but not by any DFAs?

Expressive Power of NFAs and DFAs

- Is there a language that is recognized by a DFA but not by any NFAs? No!
- Is there a language that is recognized by an NFA but not by any DFAs? No!

Main Theorem

Theorem

A language L is regular if and only if there is an NFA N such that $L(N) = L$.

Main Theorem

Theorem

A language L is regular if and only if there is an NFA N such that $L(N) = L$.
In other words:

- *For any DFA D, there is an NFA N such that $L(N) = L(D)$, and*
- *for any NFA N, there is a DFA D such that $L(D) = L(N)$.*

Converting DFAs to NFAs

Proposition

For any DFA D, there is an NFA N such that $L(N) = L(D)$.

Proof.

Is a DFA an NFA? Essentially yes! Syntactically, not quite. The formal definition of DFA has $\delta_{\mathsf{DFA}} : Q \times \Sigma \to Q$ whereas $\delta_{\mathsf{NFA}} : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$.

.

$\square$

Converting DFAs to NFAs

Proposition

For any DFA D, there is an NFA N such that $L(N) = L(D)$.

Proof.

Is a DFA an NFA? Essentially yes! Syntactically, not quite. The formal definition of DFA has $\delta_{\mathsf{DFA}} : Q \times \Sigma \to Q$ whereas $\delta_{\mathsf{NFA}} : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$.
For DFA $D = (Q, \Sigma, \delta_D, q_0, F)$, define an "equivalent" NFA $N = (Q, \Sigma, \delta_N, q_0, F)$ that has the exact same set of states, initial state and final states. Only difference is in the transition function.

$$\delta_N(q, a) = \{\delta_D(q, a)\}$$

for $a \in \Sigma$ and $\delta_N(q, \epsilon) = \emptyset$ for all $q \in Q$. □

Simulating an NFA on Your Computer

NFA Acceptance Problem

Given an NFA N and an input string w, does N accept w?

Simulating an NFA on Your Computer

NFA Acceptance Problem

Given an NFA N and an input string w, does N accept w?

How do we write a computer program to solve the NFA Acceptance problem?

Two Views of Nondeterminsm

Guessing View

At each step, the NFA “guesses” one of the choices available; the NFA will guess an “accepting sequence of choices” if such a one exists.

Parallel View

At each step the machine “forks” threads corresponding to each of the possible next states.

Two Views of Nondeterminsm

Guessing View

At each step, the NFA "guesses" one of the choices available; the NFA will guess an "accepting sequence of choices" if such a one exists.
Very useful in reasoning about NFAs and in designing NFAs.

Parallel View

At each step the machine "forks" threads corresponding to each of the possible next states.

Two Views of Nondeterminsm

Guessing View

At each step, the NFA "guesses" one of the choices available; the NFA will guess an "accepting sequence of choices" if such a one exists.

Very useful in reasoning about NFAs and in designing NFAs.

Parallel View

At each step the machine "forks" threads corresponding to each of the possible next states.

Very useful in simulating/running NFA on inputs.

Algorithm for Simulating an NFA

Algorithm

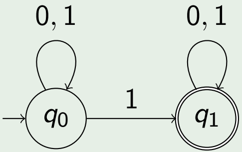
Keep track of the current state of each of the active threads.

Algorithm for Simulating an NFA

Algorithm

Keep track of the current state of each of the active threads.

Example



Example NFA N

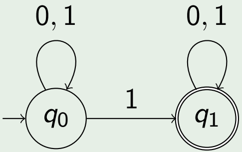
Consider the input $w = 111$. The execution (listing only the states of currently active threads) is

Algorithm for Simulating an NFA

Algorithm

Keep track of the current state of each of the active threads.

Example



Example NFA N

Consider the input $w = 111$. The execution (listing only the states of currently active threads) is

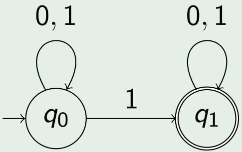
$\langle q_0 \rangle$

Algorithm for Simulating an NFA

Algorithm

Keep track of the current state of each of the active threads.

Example



Example NFA N

Consider the input $w = 111$. The execution (listing only the states of currently active threads) is

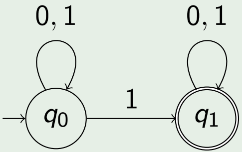
$$\langle q_0 \rangle \xrightarrow{1} \langle q_0, q_1 \rangle$$

Algorithm for Simulating an NFA

Algorithm

Keep track of the current state of each of the active threads.

Example



Example NFA N

Consider the input $w = 111$. The execution (listing only the states of currently active threads) is

$$\langle q_0 \rangle \xrightarrow{1} \langle q_0, q_1 \rangle \xrightarrow{1} \langle q_0, q_1, q_1 \rangle \xrightarrow{1} \langle q_0, q_1, q_1, q_1 \rangle$$

Algorithm

With optimizations

Observations

- Exponentially growing memory: more threads for longer inputs. Can we do better?

Algorithm

With optimizations

Observations

- Exponentially growing memory: more threads for longer inputs. Can we do better?
- Exact order of threads is not important

Algorithm
With optimizations

Observations

- Exponentially growing memory: more threads for longer inputs. Can we do better?
- Exact order of threads is not important
  - It is unimportant whether the 5^{th} thread or the 1^{st} thread is in state q.

Algorithm

With optimizations

Observations

- Exponentially growing memory: more threads for longer inputs. Can we do better?
- Exact order of threads is not important
  - It is unimportant whether the 5^{th} thread or the 1^{st} thread is in state q.
- If two threads are in the same state, then we can ignore one of the threads

Algorithm

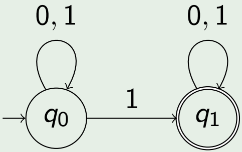
With optimizations

Observations

- Exponentially growing memory: more threads for longer inputs. Can we do better?
- Exact order of threads is not important
  - It is unimportant whether the 5^{th} thread or the 1^{st} thread is in state q.
- If two threads are in the same state, then we can ignore one of the threads
  - Threads in the same state will "behave" identically; either one of the descendent threads of both will reach a final state, or none of the descendent threads of both will reach a final state

Parsimonious Algorithm in Action

Example

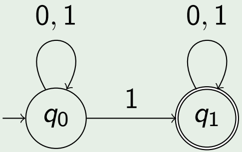


Example NFA N

Consider the input $w = 111$. The execution (listing only the states of currently active threads) is

Parsimonious Algorithm in Action

Example



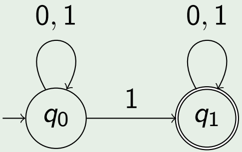
Example NFA N

Consider the input $w = 111$. The execution (listing only the states of currently active threads) is

$$\{q_0\}$$

Parsimonious Algorithm in Action

Example



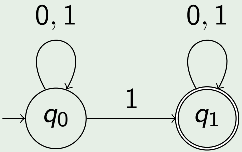
Example NFA N

Consider the input $w = 111$. The execution (listing only the states of currently active threads) is

$$\{q_0\} \xrightarrow{1} \{q_0, q_1\}$$

Parsimonious Algorithm in Action

Example



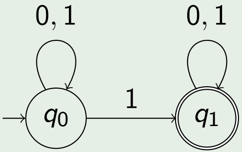
Example NFA N

Consider the input $w = 111$. The execution (listing only the states of currently active threads) is

$$\{q_0\} \xrightarrow{1} \{q_0, q_1\} \xrightarrow{1} \{q_0, q_1\}$$

Parsimonious Algorithm in Action

Example



Example NFA N

Consider the input $w = 111$. The execution (listing only the states of currently active threads) is

$$\{q_0\} \xrightarrow{1} \{q_0, q_1\} \xrightarrow{1} \{q_0, q_1\} \xrightarrow{1} \{q_0, q_1\}$$

Revisiting NFA Simulation Algorithm

- Need to keep track of the states of the active threads
  - Unordered: Without worrying about exactly which thread is in what state
  - No Duplicates: Keeping only one copy if there are multiple threads in same state

Revisiting NFA Simulation Algorithm

- Need to keep track of the states of the active threads
  - Unordered: Without worrying about exactly which thread is in what state
  - No Duplicates: Keeping only one copy if there are multiple threads in same state
- How much memory is needed?

Revisiting NFA Simulation Algorithm

- Need to keep track of the states of the active threads
  - Unordered: Without worrying about exactly which thread is in what state
  - No Duplicates: Keeping only one copy if there are multiple threads in same state
- How much memory is needed?
  - If Q is the set of states of the NFA N, then we need to keep a subset of Q!
  - Can be done in $|Q|$ bits of memory (i.e., $2^{|Q|}$ states), which is finite!!

Constructing an Equivalent DFA

- The DFA runs the simulation algorithm

Constructing an Equivalent DFA

- The DFA runs the simulation algorithm
- DFA remembers the current states of active threads without duplicates, i.e., maintains a subset of states of the NFA

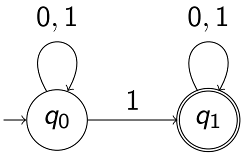
Constructing an Equivalent DFA

- The DFA runs the simulation algorithm
- DFA remembers the current states of active threads without duplicates, i.e., maintains a subset of states of the NFA
- When a new symbol is read, it updates the states of the active threads

Constructing an Equivalent DFA

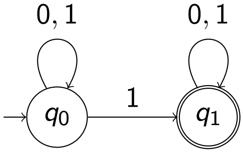
- The DFA runs the simulation algorithm
- DFA remembers the current states of active threads without duplicates, i.e., maintains a subset of states of the NFA
- When a new symbol is read, it updates the states of the active threads
- Accepts whenever one of the threads is in a final state

Example of Equivalent DFA

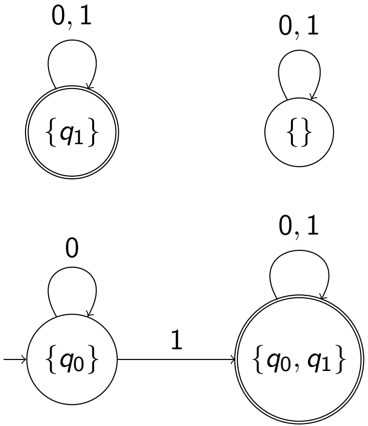


Example NFA N

Example of Equivalent DFA



Example NFA N



DFA D equivalent to N

Recall . . .

Definition

For an NFA $M = (Q, \Sigma, \delta, q_0, F)$, string w, and state $q_1 \in Q$, we say $\hat{\Delta}(q_1, w)$ to denote states of all the active threads of computation on input w from q_1.

Recall . . .

Definition

For an NFA $M = (Q, \Sigma, \delta, q_0, F)$, string w, and state $q_1 \in Q$, we say $\hat{\Delta}(q_1, w)$ to denote states of all the active threads of computation on input w from q_1. Formally,

$$\hat{\Delta}(q_1, w) = \{q \in Q \mid q_1 \xrightarrow{w}_M q\}$$

Formal Construction

Given NFA $N = (Q, \Sigma, \delta, q_0, F)$, construct DFA $\det(N) = (Q', \Sigma, \delta', q_0', F')$ as follows.

- $Q' =$
- $q_0' =$
- $F' =$

Formal Construction

Given NFA $N = (Q, \Sigma, \delta, q_0, F)$, construct DFA $\det(N) = (Q', \Sigma, \delta', q_0', F')$ as follows.

- $Q' = \mathcal{P}(Q)$
- $q_0' =$
- $F' =$

Formal Construction

Given NFA $N = (Q, \Sigma, \delta, q_0, F)$, construct DFA $\mathrm{det}(N) = (Q', \Sigma, \delta', q_0', F')$ as follows.

- $Q' = \mathcal{P}(Q)$
- $q_0' = \hat{\Delta}(q_0, \epsilon)$
- $F' =$

Formal Construction

Given NFA $N = (Q, \Sigma, \delta, q_0, F)$, construct DFA $\mathrm{det}(N) = (Q', \Sigma, \delta', q_0', F')$ as follows.

- $Q' = \mathcal{P}(Q)$
- $q_0' = \hat{\Delta}(q_0, \epsilon)$
- $F' = \{A \subseteq Q \mid A \cap F \neq \emptyset\}$

Formal Construction

Given NFA $N = (Q, \Sigma, \delta, q_0, F)$, construct DFA $\det(N) = (Q', \Sigma, \delta', q_0', F')$ as follows.

- $Q' = \mathcal{P}(Q)$
- $q_0' = \hat{\Delta}(q_0, \epsilon)$
- $F' = \{A \subseteq Q \mid A \cap F \neq \emptyset\}$
- $\delta'(\{q_1, q_2, \ldots q_k\}, a) = \hat{\Delta}(q_1, a) \cup \hat{\Delta}(q_2, a) \cup \cdots \cup \hat{\Delta}(q_k, a)$

Formal Construction

Given NFA $N = (Q, \Sigma, \delta, q_0, F)$, construct DFA $\det(N) = (Q', \Sigma, \delta', q_0', F')$ as follows.

- $Q' = \mathcal{P}(Q)$
- $q_0' = \hat{\Delta}(q_0, \epsilon)$
- $F' = \{A \subseteq Q \mid A \cap F \neq \emptyset\}$
- $\delta'(\{q_1, q_2, \ldots q_k\}, a) = \hat{\Delta}(q_1, a) \cup \hat{\Delta}(q_2, a) \cup \cdots \cup \hat{\Delta}(q_k, a)$

Formal Construction

Given NFA $N = (Q, \Sigma, \delta, q_0, F)$, construct DFA $\det(N) = (Q', \Sigma, \delta', q_0', F')$ as follows.

- $Q' = \mathcal{P}(Q)$
- $q_0' = \hat{\Delta}(q_0, \epsilon)$
- $F' = \{A \subseteq Q \mid A \cap F \neq \emptyset\}$
- $\delta'(\{q_1, q_2, \ldots q_k\}, a) = \hat{\Delta}(q_1, a) \cup \hat{\Delta}(q_2, a) \cup \cdots \cup \hat{\Delta}(q_k, a)$ or more concisely,
$$\delta'(A, a) = \bigcup_{q \in A} \hat{\Delta}(q, a)$$

Correctness

Lemma

For any NFA N, the DFA $\mathrm{det}(N)$ is equivalent to it, i.e., $L(N) = L(\mathrm{det}(N))$.

Correctness

Lemma

For any NFA N, the DFA $\mathrm{det}(N)$ is equivalent to it, i.e., $L(N) = L(\mathrm{det}(N))$.

Proof Idea

Need to show

Correctness

Lemma

For any NFA N, the DFA $\mathrm{det}(N)$ is equivalent to it, i.e., $L(N) = L(\mathrm{det}(N))$.

Proof Idea

Need to show

$$\forall w \in \Sigma^*.\ \mathrm{det}(N) \text{ accepts } w \text{ iff } N \text{ accepts } w$$

Correctness

Lemma

For any NFA N, the DFA $\mathrm{det}(N)$ is equivalent to it, i.e., $L(N) = L(\mathrm{det}(N))$.

Proof Idea

Need to show

$$\forall w \in \Sigma^*.\ \mathrm{det}(N) \text{ accepts } w \text{ iff } N \text{ accepts } w$$
$$\forall w \in \Sigma^*. \hat{\delta}(q_0', w) \in F' \text{ iff } \hat{\Delta}(q_0, w) \cap F \neq \emptyset$$

Correctness

Lemma

For any NFA N, the DFA $\mathrm{det}(N)$ is equivalent to it, i.e., $L(N) = L(\mathrm{det}(N))$.

Proof Idea

Need to show

$$\forall w \in \Sigma^*.\ \mathrm{det}(N) \text{ accepts } w \text{ iff } N \text{ accepts } w$$
$$\forall w \in \Sigma^*. \hat{\delta}(q_0', w) \in F' \text{ iff } \hat{\Delta}(q_0, w) \cap F \neq \emptyset$$
$$\forall w \in \Sigma^*.\ \text{for } A = \hat{\delta}(q_0', w),\ A \cap F \neq \emptyset \text{ iff } \hat{\Delta}(q_0, w) \cap F \neq \emptyset$$

Correctness

Lemma

For any NFA N, the DFA $\mathrm{det}(N)$ is equivalent to it, i.e., $L(N) = L(\mathrm{det}(N))$.

Proof Idea

Need to show

$$\forall w \in \Sigma^*.\ \mathrm{det}(N) \text{ accepts } w \text{ iff } N \text{ accepts } w$$
$$\forall w \in \Sigma^*. \hat{\delta}(q_0', w) \in F' \text{ iff } \hat{\Delta}(q_0, w) \cap F \neq \emptyset$$
$$\forall w \in \Sigma^*.\ \text{for } A = \hat{\delta}(q_0', w),\ A \cap F \neq \emptyset \text{ iff } \hat{\Delta}(q_0, w) \cap F \neq \emptyset$$

We will instead prove the stronger claim $\forall w \in \Sigma^*.\ \hat{\delta}(q_0', w) = A$ iff $\hat{\Delta}(q_0, w) = A$.

Correctness Proof

Lemma

$\forall w \in \Sigma^*.\ \hat{\delta}(q_0', w) = A$ *iff* $\hat{\Delta}(q_0, w) = A$.

Proof.

By induction on $|w|$

- Base Case $|w| = 0$: Then $w = \epsilon$. Now

$$\hat{\delta}(q_0', \epsilon) = q_0' \qquad \text{defn. of } \hat{\delta}$$

- Induction Hypothesis: Assume inductively that the statement holds $\forall w.\ |w| = n$

Correctness Proof

Lemma

$\forall w \in \Sigma^*.\ \hat{\delta}(q_0', w) = A$ *iff* $\hat{\Delta}(q_0, w) = A$.

Proof.

By induction on $|w|$

- Base Case $|w| = 0$: Then $w = \epsilon$. Now

$$\begin{aligned} \hat{\delta}(q_0', \epsilon) &= q_0' && \text{defn. of } \hat{\delta} \\ &= \hat{\Delta}(q_0, \epsilon) && \text{defn. of } q_0' \end{aligned}$$

- Induction Hypothesis: Assume inductively that the statement holds $\forall w.\ |w| = n$

Correctness Proof

Induction Step

Proof (contd).

- Induction Step: If $|w| = n+1$ then $w = ua$ with $|u| = n$ and $a \in \Sigma$.

$$\hat{\delta}(q_0', ua) = \delta(\hat{\delta}(q_0', u), a) \qquad \text{defn. of } \hat{\delta}$$

$\square$

Correctness Proof

Induction Step

Proof (contd).

- Induction Step: If $|w| = n + 1$ then $w = ua$ with $|u| = n$ and $a \in \Sigma$.

$$
\begin{aligned}
\hat{\delta}(q_0', ua) &= \delta(\hat{\delta}(q_0', u), a) && \text{defn. of } \hat{\delta} \\
&= \delta(\hat{\Delta}(q_0, u), a) && \text{ind. hyp.}
\end{aligned}
$$

□

Correctness Proof

Induction Step

Proof (contd).

- Induction Step: If $|w| = n+1$ then $w = ua$ with $|u| = n$ and $a \in \Sigma$.

$$\begin{aligned}
\hat{\delta}(q_0', ua) &= \delta(\hat{\delta}(q_0', u), a) && \text{defn. of } \hat{\delta} \\
&= \delta(\hat{\Delta}(q_0, u), a) && \text{ind. hyp.} \\
&= \bigcup_{q \in \hat{\Delta}(q_0, u)} \hat{\Delta}(q, a) && \text{defn. of } \delta
\end{aligned}$$

□

Correctness Proof

Induction Step

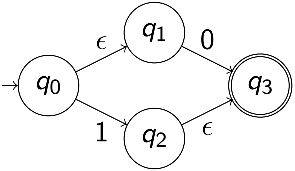
Proof (contd).

- Induction Step: If $|w| = n+1$ then $w = ua$ with $|u| = n$ and $a \in \Sigma$.

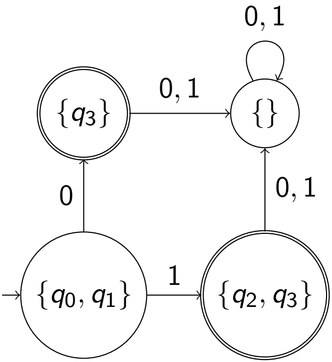
$$
\begin{aligned}
\hat{\delta}(q_0', ua) &= \delta(\hat{\delta}(q_0', u), a) && \text{defn. of } \hat{\delta} \\
&= \delta(\hat{\Delta}(q_0, u), a) && \text{ind. hyp.} \\
&= \bigcup_{q \in \hat{\Delta}(q_0, u)} \hat{\Delta}(q, a) && \text{defn. of } \delta \\
&= \hat{\Delta}(q_0, ua) && \text{prop. about } \hat{\Delta}
\end{aligned}
$$

□

Another Example



Example NFA N_ϵ



DFA D'_ϵ for N_ϵ (only relevant states)