COMP3630 / COMP6363

week 1: **Finite Automata**

This Lecture Covers Chapter 2 of HMU: Finite Automata

slides created by: Dirk Pattinson, based on material by
Peter Hoefner and Rob van Glabbeck; with improvements by Pascal Bercher

convenor & lecturer: Pascal Bercher

The Australian National University

Semester 1, 2023

COMP3630/6363: Theory of Computation

Textbook. *Introduction to Automata Theory, Languages and Computation* by John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman [HMU].

Prerequisites. Chapter 1 of HMU (sets, functions, relations, induction) (if you prefer lectures over reading, I uploaded one on YouTube)

Assessment.
- 3 assignments — each @ 12%
- 1 online quiz (1 hour) in week 4 — @ 4%
- 1 final exam — @ 60%

Labs. Participation is voluntary, but highly recommended.
- Thursday, 11 am to 1 pm and Friday, 12 pm to 2 pm
- each tutorial covers the content of the same week
- tutor of weeks 1 to 6: Adam Rowland
- tutor of weeks 7 to 12: Timothy Horscroft

Content. Languages / Automata / Computability / Complexity
This course is basically an advanced Math course.

Convenor. Pascal (Bercher), pascal.bercher@anu.edu.au

Lecturer. same!

Slides. Dirk Pattinson

CECC Class Representatives

Class Student Representation is an important component of the teaching and learning quality assurance and quality improvement processes within the ANU College of Engineering and Computer Science (CECC).

The role of Student Representatives is to provide ongoing constructive feedback on behalf of the student cohort to Course Conveners and to Associate Directors (Education) for continuous improvements to the course.

Roles and responsibilities:

- Act as the official liaison between your peers and convener.
- Be available and proactive in gathering feedback from your classmates.
- Attend regular meetings, and provide reports on course feedback to your course convener
- Close the feedback loop by reporting back to the class the outcomes of your meetings.

Why become a class representative?

- **Ensure students have a voice** to their course convener, lecturer, tutors, and College.
- **Develop skills sought by employers**, including interpersonal, dispute resolution, leadership and communication skills.
- **Become empowered**. Play an active role in determining the direction of your education.
- **Become more aware of issues influencing your University** and current issues in higher education.
- **Course design and delivery.** Help shape the delivery of your current courses, as well as future improvements for following years.

Note: Class representatives will need to be comfortable with their contact details being made available via Wattle to all students in the class.

For more information regarding roles and responsibilities, contact:

ANUSA CECC representatives: sa.cecs@anu.edu.au

Want to be a class representative? Nominate today!

Please nominate yourself to your course convener by end of Week 2, Sem 1, 2023.

You are free to nominate yourself whether you are currently on-campus or studying remotely.

Class Representatives

Interested?

- Up to two class reps per course code (COMP1600 / COMP6260)
- Self-nominate yourself by dropping Pascal an email
- Selections must be final by the end of week 2

Content of this Chapter

- Deterministic Finite Automata
- Nondeterministic Finite Automata
- NFA with ϵ-transitions
- An Equivalence among the above three.

(This was all covered in COMP1600)

Additional Reading: Chapter 2 of HMU.

Preliminary Concepts

- **Alphabet** Σ: A finite set of **symbols**, e.g.,
  - $\Sigma = \{0, 1\}$ (**binary** alphabet)
  - $\Sigma = \{a, b, \ldots, z\}$ (**Roman** alphabet)

Preliminary Concepts

- **Alphabet** Σ: A finite set of **symbols**, e.g.,
  - $\Sigma = \{0, 1\}$ (**binary** alphabet)
  - $\Sigma = \{a, b, \ldots, z\}$ (**Roman** alphabet)
- **String** (or **word**) is a finite sequence of symbols.
  Strings are usually represented without commas, e.g., 0011 instead of $(0, 0, 1, 1)$

Preliminary Concepts

- **Alphabet** Σ: A finite set of **symbols**, e.g.,
  - $\Sigma = \{0, 1\}$ (**binary** alphabet)
  - $\Sigma = \{a, b, \ldots, z\}$ (**Roman** alphabet)
- **String** (or **word**) is a finite sequence of symbols.
  Strings are usually represented without commas, e.g., 0011 instead of $(0, 0, 1, 1)$
- **Concatenation** $x \cdot y$ of strings x and y is the string xy.

(We often elide the concatenation operator and write AB for $A \cdot B$)

Preliminary Concepts

- **Alphabet** Σ: A finite set of **symbols**, e.g.,
  - $\Sigma = \{0, 1\}$ (**binary** alphabet)
  - $\Sigma = \{a, b, \dots, z\}$ (**Roman** alphabet)
- **String** (or **word**) is a finite sequence of symbols.
  Strings are usually represented without commas, e.g., 0011 instead of $(0, 0, 1, 1)$
- **Concatenation** $x \cdot y$ of strings x and y is the string xy.
  - ϵ is the identity element for concatenation, i.e., $\epsilon \cdot x = x \cdot \epsilon = x$.

(We often elide the concatenation operator and write AB for $A \cdot B$)

Preliminary Concepts

- **Alphabet** Σ: A finite set of **symbols**, e.g.,
  - $\Sigma = \{0, 1\}$ (**binary** alphabet)
  - $\Sigma = \{a, b, \ldots, z\}$ (**Roman** alphabet)
- **String** (or **word**) is a finite sequence of symbols.
  Strings are usually represented without commas, e.g., 0011 instead of $(0, 0, 1, 1)$
- **Concatenation** $x \cdot y$ of strings x and y is the string xy.
  - ϵ is the identity element for concatenation, i.e., $\epsilon \cdot x = x \cdot \epsilon = x$.
  - Concatenation of sets of strings: $A \cdot B = \{a \cdot b : a \in A, b \in B\}$

(We often elide the concatenation operator and write AB for $A \cdot B$)

Preliminary Concepts

- **Alphabet** Σ: A finite set of **symbols**, e.g.,
  - $\Sigma = \{0, 1\}$ (**binary** alphabet)
  - $\Sigma = \{a, b, \dots, z\}$ (**Roman** alphabet)
- **String** (or **word**) is a finite sequence of symbols.
  Strings are usually represented without commas, e.g., 0011 instead of $(0, 0, 1, 1)$
- **Concatenation** $x \cdot y$ of strings x and y is the string xy.
  - ϵ is the identity element for concatenation, i.e., $\epsilon \cdot x = x \cdot \epsilon = x$.
  - Concatenation of sets of strings: $A \cdot B = \{a \cdot b : a \in A, b \in B\}$
  - Concatenation of the same set: $A^2 = AA$; $A^3 = (AA)A$, etc

  (We often elide the concatenation operator and write AB for $A \cdot B$)

Preliminary Concepts

- **Alphabet** Σ: A finite set of **symbols**, e.g.,
  - $\Sigma = \{0, 1\}$ (**binary** alphabet)
  - $\Sigma = \{a, b, \ldots, z\}$ (**Roman** alphabet)
- **String** (or **word**) is a finite sequence of symbols.
  Strings are usually represented without commas, e.g., 0011 instead of $(0, 0, 1, 1)$
- **Concatenation** $x \cdot y$ of strings x and y is the string xy.
  - ϵ is the identity element for concatenation, i.e., $\epsilon \cdot x = x \cdot \epsilon = x$.
  - Concatenation of sets of strings: $A \cdot B = \{a \cdot b : a \in A, b \in B\}$
  - Concatenation of the same set: $A^2 = AA$; $A^3 = (AA)A$, etc

  (We often elide the concatenation operator and write AB for $A \cdot B$)
- Kleene-$*$ or closure operator: $A^* = \{\epsilon\} \cup A \cup A^2 \cup A^3 \cdots = \bigcup_{n \geq 0} A^n$

  (Viewing Σ as a set of length-1 strings, Σ^* is the set of all strings over Σ.)

Preliminary Concepts

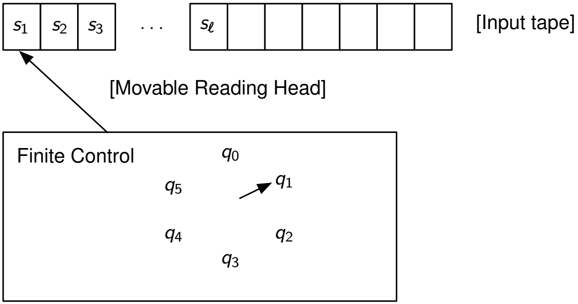
- **Alphabet** Σ: A finite set of **symbols**, e.g.,
  - $\Sigma = \{0, 1\}$ (**binary** alphabet)
  - $\Sigma = \{a, b, \ldots, z\}$ (**Roman** alphabet)
- **String** (or **word**) is a finite sequence of symbols.
  Strings are usually represented without commas, e.g., 0011 instead of $(0, 0, 1, 1)$
- **Concatenation** $x \cdot y$ of strings x and y is the string xy.
  - ϵ is the identity element for concatenation, i.e., $\epsilon \cdot x = x \cdot \epsilon = x$.
  - Concatenation of sets of strings: $A \cdot B = \{a \cdot b : a \in A, b \in B\}$
  - Concatenation of the same set: $A^2 = AA$; $A^3 = (AA)A$, etc

  (We often elide the concatenation operator and write AB for $A \cdot B$)
- Kleene-$*$ or closure operator: $A^* = \{\epsilon\} \cup A \cup A^2 \cup A^3 \cdots = \bigcup_{n \geq 0} A^n$

  (Viewing Σ as a set of length-1 strings, Σ^* is the set of all strings over Σ.)
- A **(formal) language** is a subset of Σ^*.

Deterministic Finite Automaton (DFA)

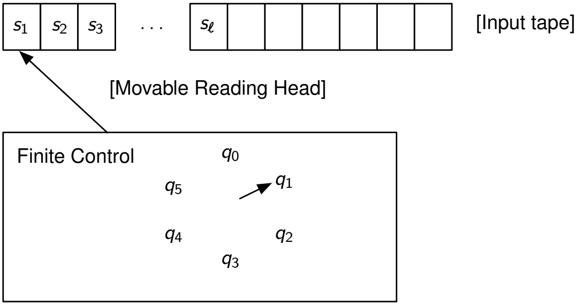
Informally:



- The device consisting of: (a) input tape; (b) reading head; and (c) finite control (Finite-state machine)

Deterministic Finite Automaton (DFA)

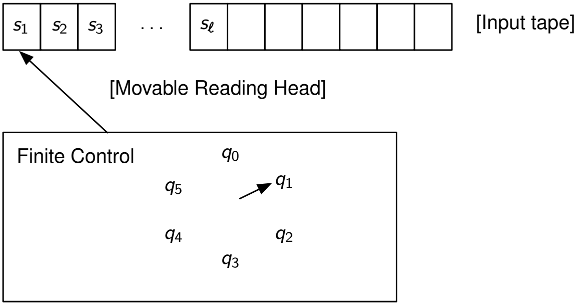
Informally:



- The device consisting of: (a) input tape; (b) reading head; and (c) finite control (Finite-state machine)
- The input is read from left to right

Deterministic Finite Automaton (DFA)

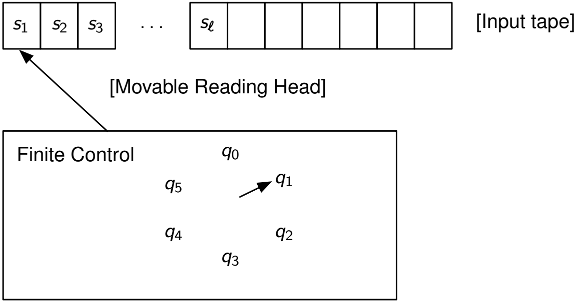
Informally:



- The device consisting of: (a) input tape; (b) reading head; and (c) finite control (Finite-state machine)
- The input is read from left to right
- Each read operation changes the internal state of the finite-state machine (FSM)

Deterministic Finite Automaton (DFA)

Informally:



- The device consisting of: (a) input tape; (b) reading head; and (c) finite control (Finite-state machine)
- The input is read from left to right
- Each read operation changes the internal state of the finite-state machine (FSM)
- Input is accepted/rejected based on the final state after reading all symbols

Deterministic Finite Automaton (DFA)

Definition: DFA

- A DFA $A = (Q, \Sigma, \delta, q_0, F)$

Deterministic Finite Automaton (DFA)

Definition: DFA

- A DFA $A = (Q, \Sigma, \delta, q_0, F)$
  - Q: A finite set (of internal states)

Deterministic Finite Automaton (DFA)

Definition: DFA

- A DFA $A = (Q, \Sigma, \delta, q_0, F)$
  - Q: A finite set (of internal states)
  - Σ: The alphabet corresponding to the input

Deterministic Finite Automaton (DFA)

Definition: DFA

- A DFA $A = (Q, \Sigma, \delta, q_0, F)$
  - Q: A finite set (of internal states)
  - Σ: The alphabet corresponding to the input
  - $\delta : Q \times \Sigma \to Q$, (Transition Function)
    (If present state is $q \in Q$, and $a \in \Sigma$ is read, the DFA moves to $\delta(q, a)$.)

Deterministic Finite Automaton (DFA)

Definition: DFA

- A DFA $A = (Q, \Sigma, \delta, q_0, F)$
  - Q: A finite set (of internal states)
  - Σ: The alphabet corresponding to the input
  - $\delta : Q \times \Sigma \rightarrow Q$, (Transition Function)
    (If present state is $q \in Q$, and $a \in \Sigma$ is read, the DFA moves to $\delta(q, a)$.)
  - q_0: The (unique) starting state of the DFA (prior to any reading). ($q_0 \in Q$)

Deterministic Finite Automaton (DFA)

Definition: DFA

- A DFA $A = (Q, \Sigma, \delta, q_0, F)$
  - Q: A finite set (of internal states)
  - Σ: The alphabet corresponding to the input
  - $\delta : Q \times \Sigma \to Q$, (Transition Function)
    (If present state is $q \in Q$, and $a \in \Sigma$ is read, the DFA moves to $\delta(q, a)$.)
  - q_0: The (unique) starting state of the DFA (prior to any reading). ($q_0 \in Q$)
  - $F \subseteq Q$ is the set of final (or accepting) states

Deterministic Finite Automaton (DFA)

Definition: DFA

- A DFA $A = (Q, \Sigma, \delta, q_0, F)$
  - Q: A finite set (of internal states)
  - Σ: The alphabet corresponding to the input
  - $\delta : Q \times \Sigma \to Q$, (Transition Function)
    (If present state is $q \in Q$, and $a \in \Sigma$ is read, the DFA moves to $\delta(q, a)$.)
  - q_0: The (unique) starting state of the DFA (prior to any reading). ($q_0 \in Q$)
  - $F \subseteq Q$ is the set of final (or accepting) states

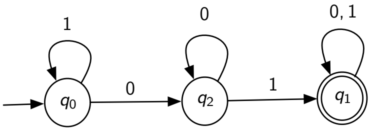
Transition Table:

	0	1
$\rightarrow q_0$	q_2	q_0
$* q_1$	q_1	q_1
q_2	q_2	q_1

$F = \{q_1\}$

$$\delta(q_0, 0) = q_2$$
$$\delta(q_0, 1) = q_0$$

Transition Diagram:



Language accepted by a DFA

- The language $L(A)$ accepted by a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is:
  - The set of all input strings that <u>move</u> the state of the DFA from q_0 to a state in F

Language accepted by a DFA

- The language $L(A)$ accepted by a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is:
  - The set of all input strings that move the state of the DFA from q_0 to a state in F
- This is formalized via the **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to Q$:

Language accepted by a DFA

- The language $L(A)$ accepted by a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is:
  - The set of all input strings that move the state of the DFA from q_0 to a state in F
- This is formalized via the **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to Q$:
- Basis:

$$\hat{\delta}(q, \epsilon) = q \quad \text{(no state change)}$$

Language accepted by a DFA

- The language $L(A)$ accepted by a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is:
  - The set of all input strings that move the state of the DFA from q_0 to a state in F
- This is formalized via the **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to Q$:
- Basis:

$$\hat{\delta}(q, \epsilon) = q \quad \text{(no state change)}$$

- Induction:

$$\hat{\delta}(q, ws) = \delta(\hat{\delta}(q, w), s) \quad \text{(process word } w\text{, then symbol } s\text{)}$$

Language accepted by a DFA

- The language $L(A)$ accepted by a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is:
  - The set of all input strings that move the state of the DFA from q_0 to a state in F
- This is formalized via the **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to Q$:
- Basis:

$$\hat{\delta}(q, \epsilon) = q \quad \text{(no state change)}$$

- Induction:

$$\hat{\delta}(q, ws) = \delta(\hat{\delta}(q, w), s) \quad \text{(process word } w\text{, then symbol } s\text{)}$$

- $L(A) :=$ all strings that take q_0 to some final state $= \{w \in \Sigma^* : \hat{\delta}(q_0, w) \in F\}$.

Language accepted by a DFA

- The language $L(A)$ accepted by a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is:
  - The set of all input strings that move the state of the DFA from q_0 to a state in F
- This is formalized via the **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to Q$:
- Basis:

$$\hat{\delta}(q, \epsilon) = q \quad \text{(no state change)}$$

- Induction:

$$\hat{\delta}(q, ws) = \delta(\hat{\delta}(q, w), s) \quad \text{(process word } w\text{, then symbol } s\text{)}$$

- $L(A) :=$ all strings that take q_0 to some final state $= \{w \in \Sigma^* : \hat{\delta}(q_0, w) \in F\}$.

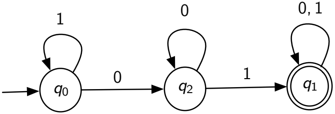
In other words:

- $\epsilon \in L(A) \Leftrightarrow q_0 \in F$
- For $k > 0$,

$$w = s_1 s_2 \cdots s_k \in L(A) \Leftrightarrow q_0 \xrightarrow{s_1} P_1 \xrightarrow{s_2} P_2 \xrightarrow{s_3} \cdots \xrightarrow{s_k} P_k \in F$$

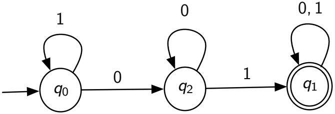
An Example

A:



An Example

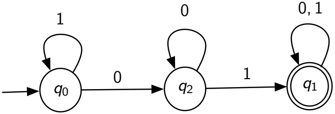
A:



- Is 00 accepted by A?

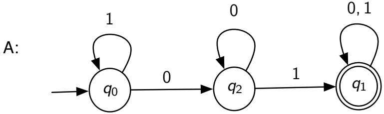
An Example

A:



- Is 00 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \notin F$

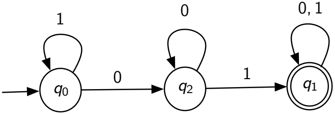
An Example



- Is 00 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \notin F$
  - Thus, $00 \notin L(A)$

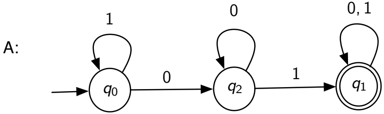
An Example

A:



- Is 00 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \notin F$
  - Thus, $00 \notin L(A)$
- Is 001 accepted by A?

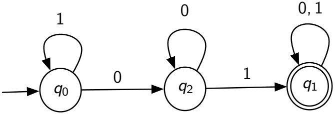
An Example



- Is 00 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \notin F$
  - Thus, $00 \notin L(A)$
- Is 001 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \xrightarrow{1} q_1 \in F$

An Example

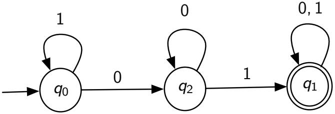
A:



- Is 00 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \notin F$
  - Thus, $00 \notin L(A)$
- Is 001 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \xrightarrow{1} q_1 \in F$
  - Thus, $001 \in L(A)$

An Example

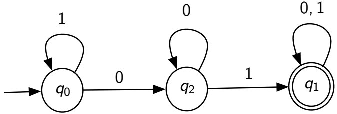
A:



- Is 00 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \notin F$
  - Thus, $00 \notin L(A)$
- Is 001 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \xrightarrow{1} q_1 \in F$
  - Thus, $001 \in L(A)$
- The only way one can reach q_1 from q_0 is if the string contains 01.

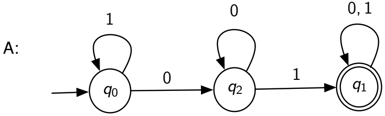
An Example

A:



- Is 00 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \notin F$
  - Thus, $00 \notin L(A)$
- Is 001 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \xrightarrow{1} q_1 \in F$
  - Thus, $001 \in L(A)$
- The only way one can reach q_1 from q_0 is if the string contains 01.
- $L(A)$ is the set of strings containing 01.

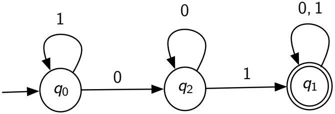
An Example



- Is 00 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \notin F$
  - Thus, $00 \notin L(A)$
- Is 001 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \xrightarrow{1} q_1 \in F$
  - Thus, $001 \in L(A)$
- The only way one can reach q_1 from q_0 is if the string contains 01.
- $L(A)$ is the set of strings containing 01.
- Remark 1: In general, each string corresponds to a unique path of states.

An Example

A:



- Is 00 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \notin F$
  - Thus, $00 \notin L(A)$
- Is 001 accepted by A?
  - $q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \xrightarrow{1} q_1 \in F$
  - Thus, $001 \in L(A)$
- The only way one can reach q_1 from q_0 is if the string contains 01.
- $L(A)$ is the set of strings containing 01.
- Remark 1: In general, each string corresponds to a unique path of states.
- Remark 2: Multiple strings can have the same path of states. For example, 0010 and 0011 have the same sequence of states.

Limitations of DFAs

- Can all languages be accepted by DFAs?

Limitations of DFAs

- Can all languages be accepted by DFAs?
  - DFAs have a finite number of states (and hence finite memory).

Limitations of DFAs

- Can all languages be accepted by DFAs?
  - DFAs have a finite number of states (and hence finite memory).
  - Given a DFA, there is always a long pattern it **cannot** 'remember' or 'track'
    - e.g., $L = \{0^n 1^n : n \in \mathbb{N}\}$ cannot be accepted by **any** DFA.

Limitations of DFAs

- Can all languages be accepted by DFAs?
  - DFAs have a finite number of states (and hence finite memory).
  - Given a DFA, there is always a long pattern it **cannot** 'remember' or 'track'
    - e.g., $L = \{0^n 1^n : n \in \mathbb{N}\}$ cannot be accepted by **any** DFA.
- Can generalize DFAs in one of many ways:

Limitations of DFAs

- Can all languages be accepted by DFAs?
  - DFAs have a finite number of states (and hence finite memory).
  - Given a DFA, there is always a long pattern it **cannot** 'remember' or 'track'
    - e.g., $L = \{0^n 1^n : n \in \mathbb{N}\}$ cannot be accepted by **any** DFA.

- Can generalize DFAs in one of many ways:
  - Allow transitions to multiple states for each symbol read.

Limitations of DFAs

- Can all languages be accepted by DFAs?
  - DFAs have a finite number of states (and hence finite memory).
  - Given a DFA, there is always a long pattern it **cannot** 'remember' or 'track'
    - e.g., $L = \{0^n 1^n : n \in \mathbb{N}\}$ cannot be accepted by **any** DFA.
- Can generalize DFAs in one of many ways:
  - Allow transitions to multiple states for each symbol read.
  - Allow transitions without reading any symbol

Limitations of DFAs

- Can all languages be accepted by DFAs?
  - DFAs have a finite number of states (and hence finite memory).
  - Given a DFA, there is always a long pattern it **cannot** 'remember' or 'track'
    - e.g., $L = \{0^n 1^n : n \in \mathbb{N}\}$ cannot be accepted by **any** DFA.
- Can generalize DFAs in one of many ways:
  - Allow transitions to multiple states for each symbol read.
  - Allow transitions without reading any symbol
  - Allow the device to have an additional tape to store symbols

Limitations of DFAs

- Can all languages be accepted by DFAs?
  - DFAs have a finite number of states (and hence finite memory).
  - Given a DFA, there is always a long pattern it **cannot** 'remember' or 'track'
    - e.g., $L = \{0^n 1^n : n \in \mathbb{N}\}$ cannot be accepted by **any** DFA.

- Can generalize DFAs in one of many ways:
  - Allow transitions to multiple states for each symbol read.
  - Allow transitions without reading any symbol
  - Allow the device to have an additional tape to store symbols
  - Allow the device to edit the input tape

Limitations of DFAs

- Can all languages be accepted by DFAs?
  - DFAs have a finite number of states (and hence finite memory).
  - Given a DFA, there is always a long pattern it **cannot** 'remember' or 'track'
    - e.g., $L = \{0^n 1^n : n \in \mathbb{N}\}$ cannot be accepted by **any** DFA.

- Can generalize DFAs in one of many ways:
  - Allow transitions to multiple states for each symbol read.
  - Allow transitions without reading any symbol
  - Allow the device to have an additional tape to store symbols
  - Allow the device to edit the input tape
  - Allow bidirectional head movement

Non-deterministic Finite Automaton (NFA)

- Allow transitions to multiple states at each symbol reading.

Non-deterministic Finite Automaton (NFA)

- Allow transitions to multiple states at each symbol reading.
- Multiple transitions allows the device to:

Non-deterministic Finite Automaton (NFA)

- Allow transitions to multiple states at each symbol reading.
- Multiple transitions allows the device to:
  - clone itself, traverse through and consider all possible parallel outcomes.

Non-deterministic Finite Automaton (NFA)

- Allow transitions to multiple states at each symbol reading.
- Multiple transitions allows the device to:
  - clone itself, traverse through and consider all possible parallel outcomes.
  - hypothesize/guess multiple eventualities concerning its input.

Non-deterministic Finite Automaton (NFA)

- Allow transitions to multiple states at each symbol reading.
- Multiple transitions allows the device to:
  - clone itself, traverse through and consider all possible parallel outcomes.
  - hypothesize/guess multiple eventualities concerning its input.
- Non-determinism seems bizarre, but aids in the simplification of describing an automaton.

Non-deterministic Finite Automaton (NFA)

- Allow transitions to multiple states at each symbol reading.
- Multiple transitions allows the device to:
  - clone itself, traverse through and consider all possible parallel outcomes.
  - hypothesize/guess multiple eventualities concerning its input.
- Non-determinism seems bizarre, but aids in the simplification of describing an automaton.

Definition: NFA

- NFA $A = (Q, \Sigma, \delta, q_0, F)$ is defined similar to a DFA with the exception of the transition function, which takes the following form.
  - $\delta : Q \times \Sigma \to 2^Q$ (Transition Function)

Non-deterministic Finite Automaton (NFA)

- Allow transitions to multiple states at each symbol reading.
- Multiple transitions allows the device to:
  - clone itself, traverse through and consider all possible parallel outcomes.
  - hypothesize/guess multiple eventualities concerning its input.
- Non-determinism seems bizarre, but aids in the simplification of describing an automaton.

Definition: NFA

- NFA $A = (Q, \Sigma, \delta, q_0, F)$ is defined similar to a DFA with the exception of the transition function, which takes the following form.
  - $\delta : Q \times \Sigma \rightarrow 2^Q$ (Transition Function)

- **Remark 1**: $\delta(q, s)$ can be a set with two or more states, or even be empty!

Non-deterministic Finite Automaton (NFA)

- Allow transitions to multiple states at each symbol reading.
- Multiple transitions allows the device to:
  - clone itself, traverse through and consider all possible parallel outcomes.
  - hypothesize/guess multiple eventualities concerning its input.
- Non-determinism seems bizarre, but aids in the simplification of describing an automaton.

Definition: NFA

- NFA $A = (Q, \Sigma, \delta, q_0, F)$ is defined similar to a DFA with the exception of the transition function, which takes the following form.
  - $\delta : Q \times \Sigma \rightarrow 2^Q$ (Transition Function)

- **Remark 1**: $\delta(q, s)$ can be a set with two or more states, or even be empty!
- **Remark 2**: If $\delta(\cdot, \cdot)$ is a singleton for all argument pairs, then NFA is a DFA. (So every DFA is trivally an NFA).

Language Accepted by an NFA

- The language accepted by an NFA is formally defined via the **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to 2^Q$:

Language Accepted by an NFA

- The language accepted by an NFA is formally defined via the **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to 2^Q$:
  - Basis:

$$\hat{\delta}(q, \epsilon) = \{q\} \quad \text{(no state change)}$$

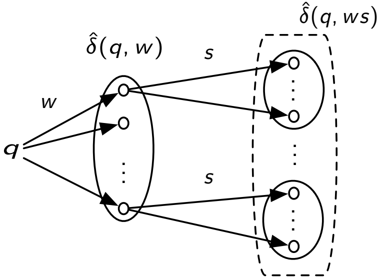
Language Accepted by an NFA

- The language accepted by an NFA is formally defined via the **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to 2^Q$:
  - Basis:

    $$\hat{\delta}(q, \epsilon) = \{q\} \quad \text{(no state change)}$$

  - Induction:

    $$\hat{\delta}(q, ws) = \bigcup_{p \in \hat{\delta}(q,w)} \delta(p, s), \; s \in \Sigma, w \in \Sigma^*.$$



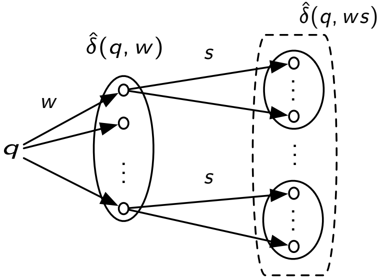
Language Accepted by an NFA

- The language accepted by an NFA is formally defined via the **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to 2^Q$:
  - Basis:

    $$\hat{\delta}(q, \epsilon) = \{q\} \quad \text{(no state change)}$$

  - Induction:

    $$\hat{\delta}(q, ws) = \bigcup_{p \in \hat{\delta}(q,w)} \delta(p, s), \;\; s \in \Sigma, w \in \Sigma^*.$$



- $L(A) := \{w \in \Sigma^* : \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$.

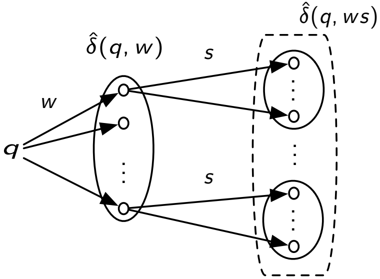
Language Accepted by an NFA

- The language accepted by an NFA is formally defined via the **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to 2^Q$:
  - Basis:

    $$\hat{\delta}(q, \epsilon) = \{q\} \quad \text{(no state change)}$$

  - Induction:

    $$\hat{\delta}(q, ws) = \bigcup_{p \in \hat{\delta}(q,w)} \delta(p, s), \;\; s \in \Sigma, w \in \Sigma^*.$$



- $L(A) := \{w \in \Sigma^* : \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$.

In other words:

- $\epsilon \in L(A) \Leftrightarrow q_0 \in F$
- For $k > 0$,

  $$w = s_1 s_2 \cdots s_k \in L(A) \Leftrightarrow \exists \text{ a path } q_0 \xrightarrow{s_1} P_1 \xrightarrow{s_2} P_2 \xrightarrow{s_3} \cdots \xrightarrow{s_k} P_k \in F$$

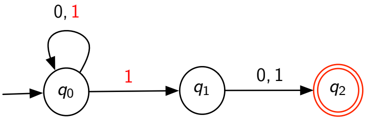
An Example

- $L(A) = \{w : \text{ penultimate}^* \text{ symbol in } w \text{ is a } 1\}$. (* = second to last!)

An Example

- $L(A) = \{w : \text{ penultimate}^* \text{ symbol in } w \text{ is a } 1\}$. (* = second to last!)

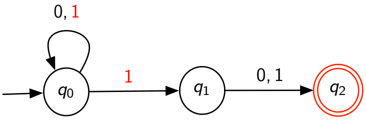
	0	1
$\rightarrow q_0$	q_0	$\{q_0, q_1\}$
q_1	q_2	q_2
$* q_2$	$\emptyset$	$\emptyset$



An Example

- $L(A) = \{w : \text{ penultimate}^* \text{ symbol in } w \text{ is a } 1\}$. (* = second to last!)

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

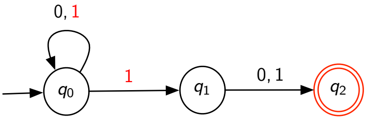


- $\hat{\delta}(q_0, 00) = \{q_0\}$ $\qquad q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0$

An Example

- $L(A) = \{w : \text{ penultimate}^* \text{ symbol in } w \text{ is a } 1\}$. (* = second to last!)

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

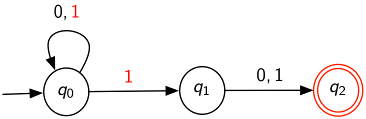


- $\hat{\delta}(q_0, 00) = \{q_0\}$ $\quad q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0$
- $\hat{\delta}(q_0, 01) = \{q_0, q_1\}$ $\quad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \quad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$

An Example

- $L(A) = \{w : \text{ penultimate}^* \text{ symbol in } w \text{ is a } 1\}$. (* = second to last!)

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

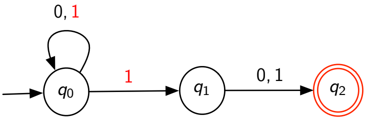


- $\hat{\delta}(q_0, 00) = \{q_0\}$ $\quad q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0$
- $\hat{\delta}(q_0, 01) = \{q_0, q_1\}$ $\quad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \quad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$
- $\hat{\delta}(q_0, 10) = \{q_0, q_2\}$ $\quad q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \quad q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2$

An Example

- $L(A) = \{w : \text{penultimate}^* \text{ symbol in } w \text{ is a } 1\}$. (* = second to last!)

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

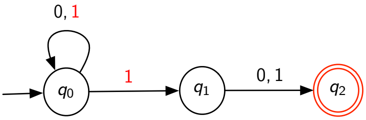


- $\hat{\delta}(q_0, 00) = \{q_0\}$ $\quad q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0$
- $\hat{\delta}(q_0, 01) = \{q_0, q_1\}$ $\quad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \quad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$
- $\hat{\delta}(q_0, 10) = \{q_0, q_2\}$ $\quad q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \quad q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2$
- $\hat{\delta}(q_0, 100) = \{q_0\}$ $\quad q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_0 \xrightarrow{0} q_0$

An Example

- $L(A) = \{w : \text{ penultimate}^* \text{ symbol in } w \text{ is a } 1\}$. ($^* =$ second to last!)

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

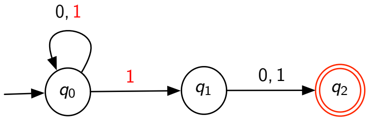


- $\hat{\delta}(q_0, 00) = \{q_0\}$ $\quad q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0$
- $\hat{\delta}(q_0, 01) = \{q_0, q_1\}$ $\quad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \quad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$
- $\hat{\delta}(q_0, 10) = \{q_0, q_2\}$ $\quad q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \quad q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2$
- $\hat{\delta}(q_0, 100) = \{q_0\}$ $\quad q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_0 \xrightarrow{0} q_0$
- An input can move the state from q_0 to q_2 only if it ends in 10 or 11.

An Example

- $L(A) = \{w : \text{ penultimate}^* \text{ symbol in } w \text{ is a } 1\}$. (* = second to last!)

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

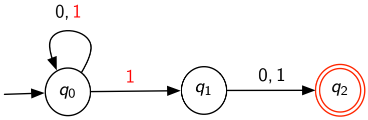


- $\hat{\delta}(q_0, 00) = \{q_0\}$ $\quad q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0$
- $\hat{\delta}(q_0, 01) = \{q_0, q_1\}$ $\quad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \quad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$
- $\hat{\delta}(q_0, 10) = \{q_0, q_2\}$ $\quad q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \quad q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2$
- $\hat{\delta}(q_0, 100) = \{q_0\}$ $\quad q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_0 \xrightarrow{0} q_0$
- An input can move the state from q_0 to q_2 only if it ends in 10 or 11.
- Each time the NFA reads a 1 (in state q_0) it considers two parallel possibilities:

An Example

- $L(A) = \{w : \text{ penultimate}^* \text{ symbol in } w \text{ is a } 1\}$. (* = second to last!)

		0	1
→	q_0	q_0	$\{q_0, q_1\}$
	q_1	q_2	q_2
*	q_2	$\emptyset$	$\emptyset$

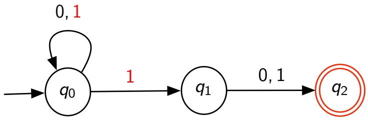


- $\hat{\delta}(q_0, 00) = \{q_0\}$ $\quad q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0$
- $\hat{\delta}(q_0, 01) = \{q_0, q_1\}$ $\quad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \qquad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$
- $\hat{\delta}(q_0, 10) = \{q_0, q_2\}$ $\quad q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \qquad q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2$
- $\hat{\delta}(q_0, 100) = \{q_0\}$ $\quad q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_0 \xrightarrow{0} q_0$
- An input can move the state from q_0 to q_2 only if it ends in 10 or 11.
- Each time the NFA reads a 1 (in state q_0) it considers two parallel possibilities:
  - the 1 is the penultimate symbol. (These paths die if the 1 is not actually the penultimate symbol)

An Example

- $L(A) = \{w : \text{ penultimate}^* \text{ symbol in } w \text{ is a } 1\}$. (* = second to last!)

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



- $\hat{\delta}(q_0, 00) = \{q_0\}$ $\qquad q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0$
- $\hat{\delta}(q_0, 01) = \{q_0, q_1\}$ $\qquad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \qquad q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$
- $\hat{\delta}(q_0, 10) = \{q_0, q_2\}$ $\qquad q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \qquad q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2$
- $\hat{\delta}(q_0, 100) = \{q_0\}$ $\qquad q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_0 \xrightarrow{0} q_0$
- An input can move the state from q_0 to q_2 only if it ends in 10 or 11.
- Each time the NFA reads a 1 (in state q_0) it considers two parallel possibilities:
  - the 1 is the penultimate symbol. (These paths die if the 1 is not actually the penultimate symbol)
  - the 1 is not the penultimate symbol.

Is Non-determinism Better?

- Non-determinism was introduced to increase the computational power.

Is Non-determinism Better?

- Non-determinism was introduced to increase the computational power.
- So is there a language L that is accepted by an NDA, but not by any DFA?

Is Non-determinism Better?

- Non-determinism was introduced to increase the computational power.
- So is there a language L that is accepted by an NDA, but not by any DFA?

Theorem 2.4.1

Every Language L that is accepted by an NFA is also accepted by some DFA.

Is Non-determinism Better?

Proof of Theorem 2.4.1

- Let NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ generate the given language L

Is Non-determinism Better?

Proof of Theorem 2.4.1

- Let NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ generate the given language L
- **Idea:** Devise a DFA D such that at any time instant the state of the DFA is the set of all states that NFA N can be in.

Is Non-determinism Better?

Proof of Theorem 2.4.1

- Let NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ generate the given language L
- **Idea:** Devise a DFA D such that at any time instant the state of the DFA is the set of all states that NFA N can be in.
- Define DFA $D = (Q_D, \Sigma, \delta_D, q_{D,0}, F_D)$ from N using the following **subset construction**:

Is Non-determinism Better?

Proof of Theorem 2.4.1

- Let NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ generate the given language L
- **Idea:** Devise a DFA D such that at any time instant the state of the DFA is the set of all states that NFA N can be in.
- Define DFA $D = (Q_D, \Sigma, \delta_D, q_{D,0}, F_D)$ from N using the following **subset construction**:

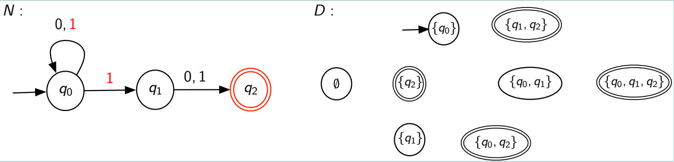
$$Q_D = 2^{Q_N} \qquad q_{D,0} = \{q_0\} \qquad F_D = \{S \subseteq Q_N : S \cap F_N \neq \emptyset\}$$

Is Non-determinism Better?

Proof of Theorem 2.4.1

- Let NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ generate the given language L
- **Idea:** Devise a DFA D such that at any time instant the state of the DFA is the set of all states that NFA N can be in.
- Define DFA $D = (Q_D, \Sigma, \delta_D, q_{D,0}, F_D)$ from N using the following **subset construction**:

$$Q_D = 2^{Q_N} \qquad q_{D,0} = \{q_0\} \qquad F_D = \{S \subseteq Q_N : S \cap F_N \neq \emptyset\}$$



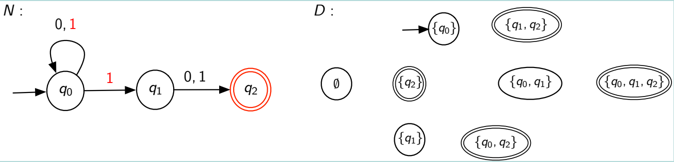
(transitions will be shown later)

Is Non-determinism Better?

Proof of Theorem 2.4.1

- Let NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ generate the given language L
- **Idea:** Devise a DFA D such that at any time instant the state of the DFA is the set of all states that NFA N can be in.
- Define DFA $D = (Q_D, \Sigma, \delta_D, q_{D,0}, F_D)$ from N using the following **subset construction**:

$$Q_D = 2^{Q_N} \qquad q_{D,0} = \{q_0\} \qquad F_D = \{S \subseteq Q_N : S \cap F_N \neq \emptyset\}$$



(transitions will be shown later)

- Hence, $\epsilon \in L(N) \;\Leftrightarrow\; q_0 \in F_N \;\Leftrightarrow\; \{q_0\} \in F_D \;\Leftrightarrow\; \epsilon \in L(D)$

Is Non-determinism Better?

Proof of Theorem 2.4.1

- To define $\delta_D(P, s)$ for each $P \subseteq Q$ and $s \in \Sigma$:

Is Non-determinism Better?

Proof of Theorem 2.4.1

- To define $\delta_D(P, s)$ for each $P \subseteq Q$ and $s \in \Sigma$:
  - Assume NFA N is simultaneously in all states of P

Is Non-determinism Better?

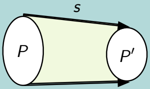
Proof of Theorem 2.4.1

- To define $\delta_D(P, s)$ for each $P \subseteq Q$ and $s \in \Sigma$:
  - Assume NFA N is simultaneously in all states of P
  - Let P' be the states to which N can transition from states in P upon reading s

Is Non-determinism Better?

Proof of Theorem 2.4.1

- To define $\delta_D(P, s)$ for each $P \subseteq Q$ and $s \in \Sigma$:
  - Assume NFA N is simultaneously in all states of P
  - Let P' be the states to which N can transition from states in P upon reading s
  - Set $\delta_D(P, s) := P' = \bigcup_{p \in P} \delta_N(p, s)$.

N: 

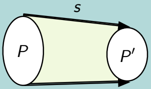
$\Leftrightarrow$ D : $P \xrightarrow{s} P'$

Is Non-determinism Better?

Proof of Theorem 2.4.1

- To define $\delta_D(P, s)$ for each $P \subseteq Q$ and $s \in \Sigma$:
  - Assume NFA N is simultaneously in all states of P
  - Let P' be the states to which N can transition from states in P upon reading s
  - Set $\delta_D(P, s) := P' = \bigcup_{p \in P} \delta_N(p, s)$.

N:



$\Leftrightarrow$

D :

$P \xrightarrow{s} P'$

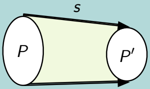
- By Induction: $\hat{\delta}_N(q_0, w) = \hat{\delta}_D(\{q_0\}, w)$ for all $w \in \Sigma^*$

Is Non-determinism Better?

Proof of Theorem 2.4.1

- To define $\delta_D(P, s)$ for each $P \subseteq Q$ and $s \in \Sigma$:
  - Assume NFA N is simultaneously in all states of P
  - Let P' be the states to which N can transition from states in P upon reading s
  - Set $\delta_D(P, s) := P' = \bigcup_{p \in P} \delta_N(p, s)$.

N:



$\Leftrightarrow$

D : $P \xrightarrow{s} P'$

- By Induction: $\hat{\delta}_N(q_0, w) = \hat{\delta}_D(\{q_0\}, w)$ for all $w \in \Sigma^*$
  - Basis: Let $s \in \Sigma$

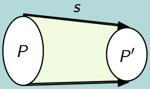
$$\hat{\delta}_N(q_0, \epsilon) \stackrel{def}{=} \{q_0\} \stackrel{def}{=} \hat{\delta}_D(\{q_0\}, \epsilon)$$

Is Non-determinism Better?

Proof of Theorem 2.4.1

- To define $\delta_D(P, s)$ for each $P \subseteq Q$ and $s \in \Sigma$:
  - Assume NFA N is simultaneously in all states of P
  - Let P' be the states to which N can transition from states in P upon reading s
  - Set $\delta_D(P, s) := P' = \bigcup_{p \in P} \delta_N(p, s)$.

N:



$\Leftrightarrow$

D : $P \xrightarrow{s} P'$

- By Induction: $\hat{\delta}_N(q_0, w) = \hat{\delta}_D(\{q_0\}, w)$ for all $w \in \Sigma^*$
  - Basis: Let $s \in \Sigma$

$$\hat{\delta}_N(q_0, \epsilon) \overset{def}{=} \{q_0\} \overset{def}{=} \hat{\delta}_D(\{q_0\}, \epsilon)$$

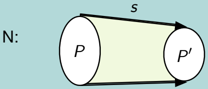
  - Induction: assume $\hat{\delta}_N(q_0, w) = \hat{\delta}_D(\{q_0\}, w)$ for $w \in \Sigma^*$

$$\hat{\delta}_N(q_0, ws) \overset{def}{=} \bigcup_{p \in \hat{\delta}_N(q_0, w)} \delta_N(p, s) \overset{ind}{=} \bigcup_{p \in \hat{\delta}_{\color{red}{D}}(\{q_0\}, w)} \delta_N(p, s) \overset{def}{=} \delta_D(\hat{\delta}_D(\{q_0\}, w), s) \overset{def}{=} \hat{\delta}_D(\{q_0\}, ws)$$

Is Non-determinism Better?

Proof of Theorem 2.4.1

- To define $\delta_D(P, s)$ for each $P \subseteq Q$ and $s \in \Sigma$:
  - Assume NFA N is simultaneously in all states of P
  - Let P' be the states to which N can transition from states in P upon reading s
  - Set $\delta_D(P, s) := P' = \bigcup_{p \in P} \delta_N(p, s)$.



$\Leftrightarrow$ $D: \quad P \xrightarrow{s} P'$

- By Induction: $\hat{\delta}_N(q_0, w) = \hat{\delta}_D(\{q_0\}, w)$ for all $w \in \Sigma^*$
  - Basis: Let $s \in \Sigma$

$$\hat{\delta}_N(q_0, \epsilon) \stackrel{def}{=} \{q_0\} \stackrel{def}{=} \hat{\delta}_D(\{q_0\}, \epsilon)$$

  - Induction: assume $\hat{\delta}_N(q_0, w) = \hat{\delta}_D(\{q_0\}, w)$ for $w \in \Sigma^*$

$$\hat{\delta}_N(q_0, ws) \stackrel{def}{=} \bigcup_{p \in \hat{\delta}_N(q_0, w)} \delta_N(p, s) \stackrel{ind}{=} \bigcup_{p \in \hat{\delta}_D(\{q_0\}, w)} \delta_N(p, s) \stackrel{def}{=} \delta_D(\hat{\delta}_D(\{q_0\}, w), s) \stackrel{def}{=} \hat{\delta}_D(\{q_0\}, ws)$$

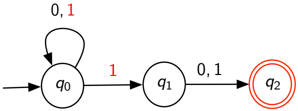
- Thus, $\hat{\delta}_N(q_0, \cdot) = \hat{\delta}_D(\{q_0\}, \cdot)$, and hence the languages have to be identical.

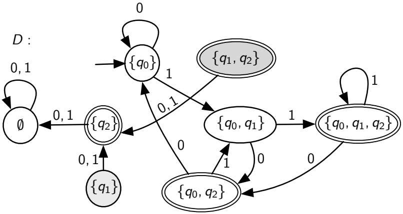
Comments about the Subset Construction Method

- Generally, the DFA constructed using subset construction has 2^n states ($n =$ number of states in the NFA).

Comments about the Subset Construction Method

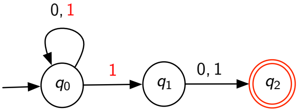
- Generally, the DFA constructed using subset construction has 2^n states (n = number of states in the NFA).
- Not all states are reachable! (see example below)

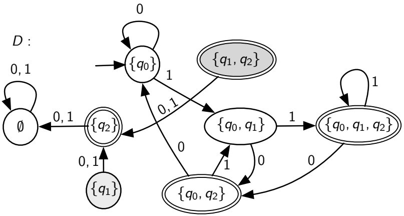




Comments about the Subset Construction Method

- Generally, the DFA constructed using subset construction has 2^n states (n = number of states in the NFA).
- Not all states are reachable! (see example below)
- The state corresponding to the empty set is **never** a final state.





ϵ-Transitions

- State transitions occur without reading any symbols.

ϵ-Transitions

- State transitions occur without reading any symbols.

Definition: ϵ-transitions

An ϵ-Nondeterministic Finite Automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ defined similar to a DFA with the exception of the transition function, which is defined to be:

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$$

ϵ-Transitions

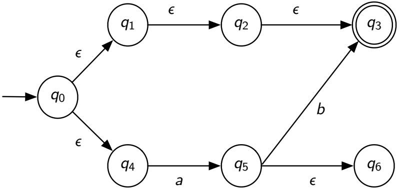
- State transitions occur without reading any symbols.

Definition: ϵ-transitions

An ϵ-Nondeterministic Finite Automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ defined similar to a DFA with the exception of the transition function, which is defined to be:

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$$

- An Example:



	ϵ	a	b
$\rightarrow q_0$	$\{q_1, q_4\}$	$\emptyset$	$\emptyset$
q_1	$\{q_2\}$	$\emptyset$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\emptyset$
$* q_3$	$\emptyset$	$\emptyset$	$\emptyset$
q_4	$\emptyset$	$\{q_5\}$	$\emptyset$
q_5	$\{q_6\}$	$\emptyset$	$\{q_3\}$
q_6	$\emptyset$	$\emptyset$	$\emptyset$

ϵ-Transitions

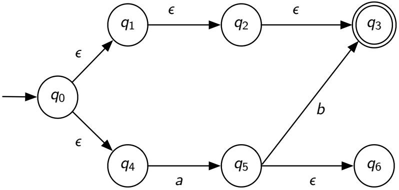
- State transitions occur without reading any symbols.

Definition: ϵ-transitions

An ϵ-Nondeterministic Finite Automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ defined similar to a DFA with the exception of the transition function, which is defined to be:

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$$

- An Example:



	ϵ	a	b
$\to q_0$	$\{q_1, q_4\}$	$\emptyset$	$\emptyset$
q_1	$\{q_2\}$	$\emptyset$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\emptyset$
$* q_3$	$\emptyset$	$\emptyset$	$\emptyset$
q_4	$\emptyset$	$\{q_5\}$	$\emptyset$
q_5	$\{q_6\}$	$\emptyset$	$\{q_3\}$
q_6	$\emptyset$	$\emptyset$	$\emptyset$

- Without reading any input symbols, the state of the ϵ-NFA can transition:

From q_0 to q_1, q_4, q_2, or q_3. From q_1 to q_2, or q_3.

From q_2 to q_3. From q_5 to q_6.

Language Accepted by an ϵ-NFA

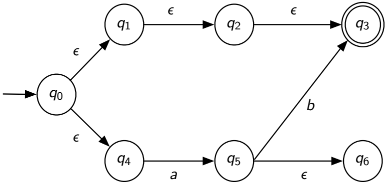
- ϵ-**closure** of a state

Language Accepted by an ϵ-NFA

- ϵ-**closure** of a state
  - $\texttt{ECLOSE}(q)$ = all states that are reachable from q by ϵ-transitions alone.

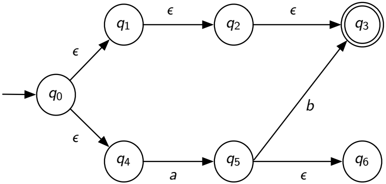
Language Accepted by an ϵ-NFA

- ϵ-**closure** of a state
  - $\texttt{ECLOSE}(q)$ = all states that are reachable from q by ϵ-transitions alone.



Language Accepted by an ϵ-NFA

- ϵ-**closure** of a state
  - $\texttt{ECLOSE}(q)$ = all states that are reachable from q by ϵ-transitions alone.



$$
\begin{aligned}
\texttt{ECLOSE}(q_0) &= \{q_0, q_1, q_4, q_2, q_3\} \\
\texttt{ECLOSE}(q_1) &= \{q_1, q_2, q_3\} \\
\texttt{ECLOSE}(q_2) &= \{q_2, q_3\} \\
\texttt{ECLOSE}(q_3) &= \{q_3\} \\
\texttt{ECLOSE}(q_4) &= \{q_4\} \\
\texttt{ECLOSE}(q_5) &= \{q_5, q_6\} \\
\texttt{ECLOSE}(q_6) &= \{q_6\}
\end{aligned}
$$

Language Accepted by an ϵ-NFA

Given ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ define **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to 2^Q$ by induction:

Language Accepted by an ϵ-NFA

Given ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ define **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to 2^Q$ by induction:

- Basis:

$$\hat{\delta}(q, \epsilon) = \mathtt{ECLOSE}(q)$$

$$q \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} q' \qquad \epsilon = \epsilon^2 = \epsilon^3 = \cdots$$

Language Accepted by an ϵ-NFA

Given ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ define **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to 2^Q$ by induction:

- Basis:

$$\hat{\delta}(q, \epsilon) = \mathtt{ECLOSE}(q)$$

$$q \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} q' \qquad \epsilon = \epsilon^2 = \epsilon^3 = \cdots$$

$$\hat{\delta}(q, s) = \bigcup_{p \in \mathtt{ECLOSE}(q)} \left(\bigcup_{p' \in \delta(p,s)} \mathtt{ECLOSE}(p') \right) \qquad [s = \underbrace{\epsilon \cdots \epsilon}_{\text{finitely many}} \; s \; \underbrace{\epsilon \cdots \epsilon}_{\text{finitely many}}]$$

$$q \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} q' \xrightarrow{s} p' \xrightarrow{\epsilon} p_1 \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} p$$

Language Accepted by an ϵ-NFA

Given ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ define **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to 2^Q$ by induction:

- Basis:

$$\hat{\delta}(q, \epsilon) = \mathtt{ECLOSE}(q)$$

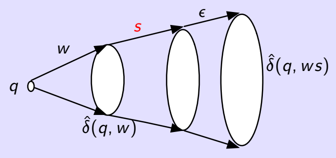
$$q \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} q' \qquad \epsilon = \epsilon^2 = \epsilon^3 = \cdots$$

$$\hat{\delta}(q, s) = \bigcup_{p \in \mathtt{ECLOSE}(q)} \left(\bigcup_{p' \in \delta(p,s)} \mathtt{ECLOSE}(p') \right) \qquad [s = \underbrace{\epsilon \cdots \epsilon}_{\text{finitely many}} \; s \; \underbrace{\epsilon \cdots \epsilon}_{\text{finitely many}}]$$

$$q \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} q' \xrightarrow{\color{red}{s}} p' \xrightarrow{\epsilon} p_1 \xrightarrow{\epsilon} \cdots \xrightarrow{} p$$

- Induction:

$$\hat{\delta}(q, ws) = \bigcup_{p \in \hat{\delta}(q,w)} \left(\bigcup_{p' \in \delta(p,s)} \color{red}{\mathtt{ECLOSE}}(p') \right)$$



Language Accepted by an ϵ-NFA

Given ϵ-NFA $N = (Q, \Sigma, \delta, q_0, F)$ define **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to 2^Q$ by induction:

- Basis:

$$\hat{\delta}(q, \epsilon) = \texttt{ECLOSE}(q)$$

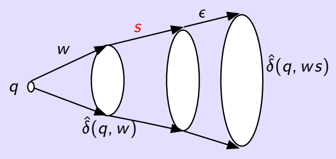
$$q \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} q' \qquad \epsilon = \epsilon^2 = \epsilon^3 = \cdots$$

$$\hat{\delta}(q, s) = \bigcup_{p \in \texttt{ECLOSE}(q)} \left(\bigcup_{p' \in \delta(p, s)} \texttt{ECLOSE}(p') \right) \qquad [s = \underbrace{\epsilon \cdots \epsilon}_{\text{finitely many}} \; s \; \underbrace{\epsilon \cdots \epsilon}_{\text{finitely many}}]$$

$$q \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} q' \xrightarrow{s} p' \xrightarrow{\epsilon} p_1 \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} p$$

- Induction:

$$\hat{\delta}(q, ws) = \bigcup_{p \in \hat{\delta}(q, w)} \left(\bigcup_{p' \in \delta(p, s)} \texttt{ECLOSE}(p') \right)$$



- w $\in L(N)$ if and only if $\hat{\delta}(q_0, \mathrm{w}) \cap F \neq \emptyset$

Language Accepted by an ϵ-NFA

- $\mathrm{w} \in L(N)$ if and only if $\hat{\delta}(q_0, \mathrm{w}) \cap F \neq \emptyset$

Language Accepted by an ϵ-NFA

- $w \in L(N)$ if and only if $\hat{\delta}(q_0, w) \cap F \neq \emptyset$
- In other words:
  - $\epsilon \in L(N) \Leftrightarrow \texttt{ECLOSE}(q_0) \cap F \neq \emptyset$

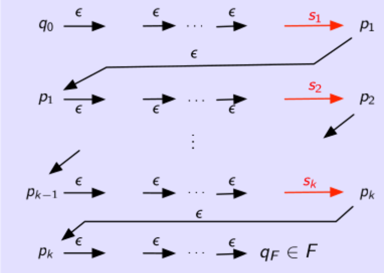
$$q_0 \xrightarrow{\epsilon} p_1 \xrightarrow{\epsilon} \dots \xrightarrow{\epsilon} p_r \in F$$

Language Accepted by an ϵ-NFA

- w $\in L(N)$ if and only if $\hat{\delta}(q_0, \mathrm{w}) \cap F \neq \emptyset$
- In other words:
  - $\epsilon \in L(N) \Leftrightarrow \mathtt{ECLOSE}(q_0) \cap F \neq \emptyset$

$$q_0 \xrightarrow{\epsilon} p_1 \xrightarrow{\epsilon} \dots \xrightarrow{\epsilon} p_r \in F$$

  - For $k > 0$, $w = s_1 s_2 \dots s_k \in L(N) \Leftrightarrow \exists$ a path such as the following:



Do ϵ-NFAs Accept More Languages?

Theorem 2.5.1

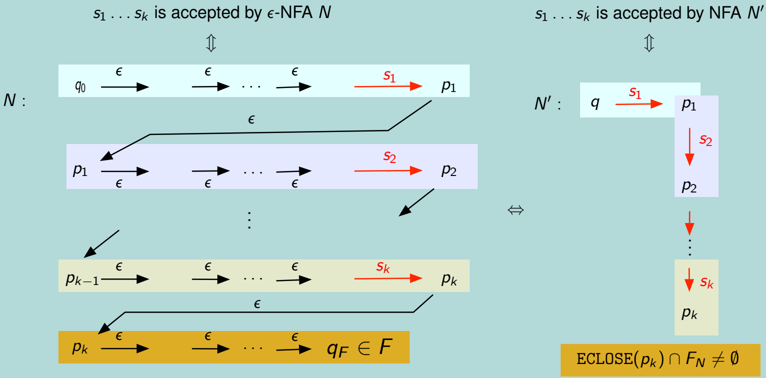
Every Language L that is accepted by an ϵ-NFA is also accepted by some DFA.

Do ϵ-NFAs Accept More Languages?

Theorem 2.5.1

Every Language L that is accepted by an ϵ-NFA is also accepted by some DFA.

Proof of Theorem 2.5.1 (Abstract idea)



Do ϵ-NFAs Accept More Languages?

Proof of Theorem 2.5.1 (Cont'd)

- Given L that is accepted by some ϵ-NFA, we must find an NFA that accepts L. ([NFA to DFA conversion can then be done as in Theorem 2.4.1].

Do ϵ-NFAs Accept More Languages?

Proof of Theorem 2.5.1 (Cont'd)

- Given L that is accepted by some ϵ-NFA, we must find an NFA that accepts L. ([NFA to DFA conversion can then be done as in Theorem 2.4.1].
- Let ϵ-NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ accept L.

Do ϵ-NFAs Accept More Languages?

Proof of Theorem 2.5.1 (Cont'd)

- Given L that is accepted by some ϵ-NFA, we must find an NFA that accepts L. ([NFA to DFA conversion can then be done as in Theorem 2.4.1].
- Let ϵ-NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ accept L.
- Let us devise NFA $N' = (Q_{N'}, \Sigma, \delta_{N'}, q_0', F_{N'})$ as follows:

Do ϵ-NFAs Accept More Languages?

Proof of Theorem 2.5.1 (Cont'd)

- Given L that is accepted by some ϵ-NFA, we must find an NFA that accepts L. ([NFA to DFA conversion can then be done as in Theorem 2.4.1].
- Let ϵ-NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ accept L.
- Let us devise NFA $N' = (Q_{N'}, \Sigma, \delta_{N'}, q'_0, F_{N'})$ as follows:

$$Q_{N'} = Q_N \qquad q'_0 = q_0 \qquad F'_N = \{q \in Q_N : \texttt{ECLOSE}(q) \cap F_N \neq \emptyset\}$$

$$\delta_{N'} : Q_{N'} \times \Sigma \to 2^{Q_{N'}} \text{ defined by: } \delta_{N'}(q, s) = \bigcup_{p \in \texttt{ECLOSE}(q)} \delta(p, s)$$

Do ϵ-NFAs Accept More Languages?

Proof of Theorem 2.5.1 (Cont'd)

- Given L that is accepted by some ϵ-NFA, we must find an NFA that accepts L. ([NFA to DFA conversion can then be done as in Theorem 2.4.1].
- Let ϵ-NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ accept L.
- Let us devise NFA $N' = (Q_{N'}, \Sigma, \delta_{N'}, q_0', F_{N'})$ as follows:

$$Q_{N'} = Q_N \qquad q_0' = q_0 \qquad F_N' = \{q \in Q_N : \mathtt{ECLOSE}(q) \cap F_N \neq \emptyset\}$$

$$\delta_{N'} : Q_{N'} \times \Sigma \to 2^{Q_{N'}} \text{ defined by: } \delta_{N'}(q,s) = \bigcup_{p \in \mathtt{ECLOSE}(q)} \delta(p,s)$$

$$N: \quad q \xrightarrow{\epsilon} \quad \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} p \xrightarrow{s} p'$$

N : q can transition to p' after a few ϵ-transitions, and a single read of $s \in \Sigma$.

$$\Updownarrow$$

$$N': \quad q \xrightarrow{s} p'$$

N': q can transition to p' after reading s.

To Summarize...

Languages accepted by DFAs	=	Languages accepted by NFAs	=	Languages accepted by ϵ-NFAs

- Allowing non-determinism and/or ϵ-transitions does not improve the language acceptance power of (finite) automata.