

Automata Theory, Computability and Complexity

Mridul Aanjaneya



Stanford University

June 26, 2012

- **Instructor:** Mridul Aanjaneya
Office Hours: 2:00PM - 4:00PM, Gates 206 (Mon).
- **Course Assistant:** Sean Kandel
Office Hours: 6:00PM - 8:00PM, Gates 372 (Tue/Thu).
- **Textbook:** Michael T. Sipser, *Introduction to the Theory of Computation* (second edition).
- **Website:** <http://www.stanford.edu/class/cs154/>
- **Discussion:** Lore (<http://www.lore.com/>)
Access code: R4ML4Z

Course Requirements

- Homework
- Midterm
- Final (optional)

Note:

- Final grade will be computed from Homework (60%) and **best** of Midterm/Final (40%).
- Projected grade will be out **one week before** the Final, taking Homework (60%) and Midterm (40%).
 - Can skip Final if satisfied with grade.
- Can collaborate on homeworks, but separate write-up per person and must mention collaborators.
- **4** homeworks, **2 free late days**, use them however you want.
 - No late days for **final** homework.

Why Study Automata?

- Regular expressions:
 - are used in many systems, e.g., UNIX.
 - DTD's describe XML tags with RE like format.
- Finite automata model protocols, electronic circuits.
 - Theory is used in **model-checking**.
- Context free languages:
 - are used as syntax descriptors for PL's, parsers (YACC).
 - have an important role in describing natural languages.
 - find use in **procedural modeling**.

- Deterministic/Nondeterministic finite automata.
- Regular expressions.
- Decision/closure properties of regular languages.
- Context-free languages. Parse trees. Normal forms.
- Pushdown automata. Equivalence of CFG's and PDA's.
- Pumping lemma for CFL's. Properties of CFL's.
- Enumerations, Turing machines.
- Undecidable problems. NP-completeness.
- Satisfiability. Cook's theorem.

Main Question:

What are the fundamental **capabilities** and **limitations** of computers?

- Dates back to the **1930s**.
- **Different** interpretations in all three areas.

Question

Why are some problems **hard** and others **easy**?

- **Subset sum:** Given a set of integers, does any non-empty subset of them add up to zero?
- So far only a **classification** according to hardness.
- Can help alter the **root** of difficulty, settle for an **approximate** solution, some problems are hard only in the **worst** case.

Question

Which problems are **solvable**?

- **Halting problem:** Given an arbitrary computer program, decide if it finishes or continues running forever.
- Computability and complexity are **closely** related.

- Deals with different **models** of computation.



Preliminaries: Mathematical Induction

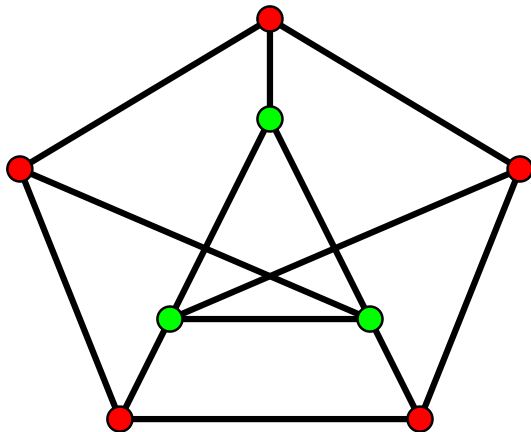


Example

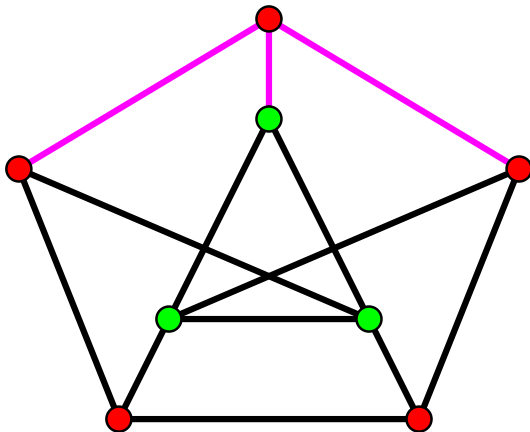
Prove that $1^2 + 2^2 + 3^2 + \dots + n^2$ is $\frac{n(n+1)(2n+1)}{6}$.

- **Basis (n=1):** $\frac{1(1+1)(2 \cdot 1 + 1)}{6} = \frac{2 \cdot 3}{6} = 1 = 1^2$.
- **Induction:** Let $\mathcal{P}(n) := 1^2 + \dots + n^2$ is $\frac{n(n+1)(2n+1)}{6}$.
$$\begin{aligned}\mathcal{P}(n+1) &= 1^2 + \dots + n^2 + (n+1)^2 = \frac{n(n+1)(2n+1)}{6} + (n+1)^2 \\ &= (n+1) \left\{ \frac{n(2n+1)+6n+6}{6} \right\} = (n+1) \left\{ \frac{2n^2+7n+6}{6} \right\} \\ &= (n+1) \left\{ \frac{2n^2+4n+3n+6}{6} \right\} = \frac{(n+1)(n+2)(2n+3)}{6}\end{aligned}$$
- Since $\mathcal{P}(1)$ is true and $\mathcal{P}(n) \Rightarrow \mathcal{P}(n+1)$, we conclude that $\mathcal{P}(n)$ is true for all $n \geq 1$.

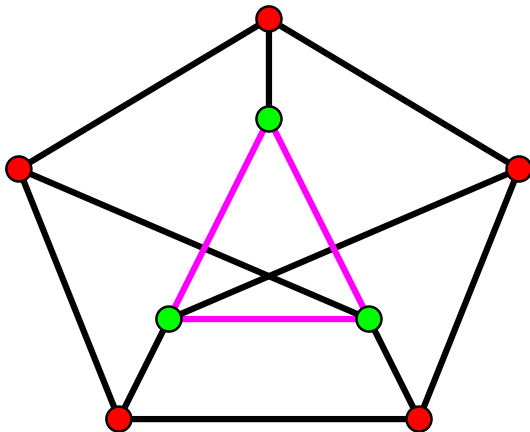
- Undirected graph



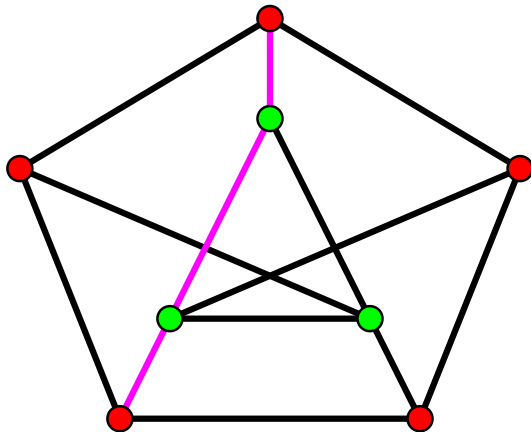
- Degree of a vertex



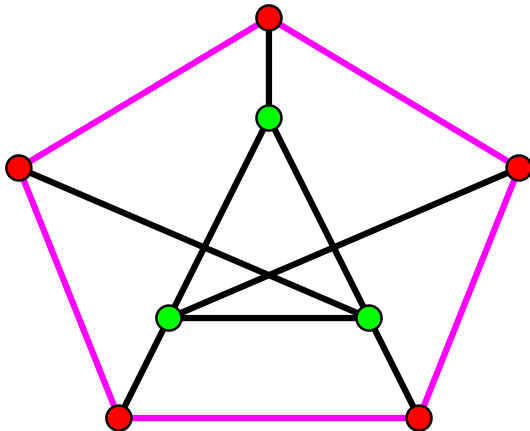
- Subgraph



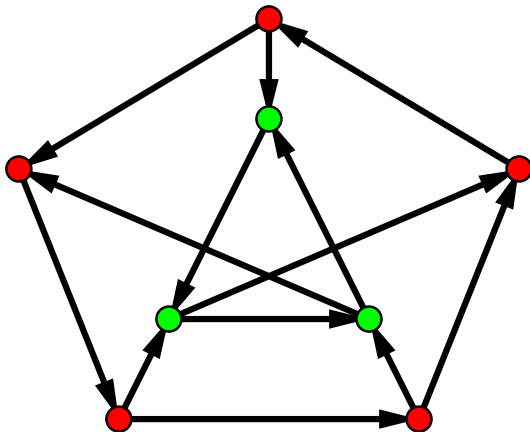
- Path



- Cycle

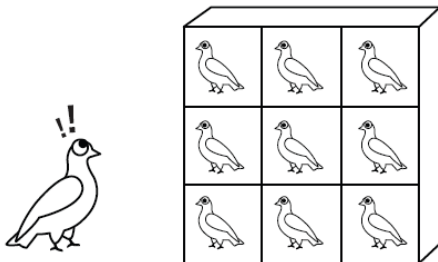


- Directed graph



Preliminaries: Pigeonhole Principle

THE PIGEONHOLE PRINCIPLE

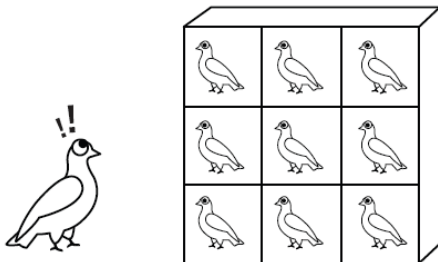


Pigeonhole Principle

If there are $n + 1$ pigeons and n pigeonholes, then some pigeonhole must contain at least two pigeons.

Preliminaries: Pigeonhole Principle

THE PIGEONHOLE PRINCIPLE



Generalized Pigeonhole Principle

If there are $kn + 1$ pigeons and n pigeonholes, then some pigeonhole must contain at least $k + 1$ pigeons.

Preliminaries: Pigeonhole Principle

Example 1

Given twelve integers, show that two of them can be chosen whose difference is divisible by 11.

Example 2

Twenty-five crates of apples are delivered to a store. The apples are of three different sorts, and all the apples in each crate are of the same sort. Show that among these crates there are at least nine containing the same sort of apple.

Example 3

Show that in any group of five people, there are two who have an identical number of friends within the group.

Alphabet

- An **alphabet** is any finite set of symbols.
 - Some examples include: ASCII, $\{0,1\}$ (binary), $\{a,b,c\}$.
- The set of **strings** over an alphabet Σ is the set of lists, each element of which is a member of Σ .
 - **Note:** Strings are shown with **no commas**, e.g., abc, 01101.
- Σ^* denotes this set of strings.
- ε stands for the **empty** string (string of **length 0**).

Example:

- For the alphabet $\Sigma = \{0, 1\}$, the set of strings in Σ^* is:
 - $\{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \dots\}$
- For the alphabet $\Sigma = \{a, b, c\}$, the set of strings in Σ^* is:
 - $\{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, \dots\}$

Alphabet

- An **alphabet** is any finite set of symbols.
 - Some examples include: ASCII, $\{0,1\}$ (binary), $\{a,b,c\}$.
- The set of **strings** over an alphabet Σ is the set of lists, each element of which is a member of Σ .
 - **Note:** Strings are shown with **no commas**, e.g., abc, 01101.
- Σ^* denotes this set of strings.
- ε stands for the **empty** string (string of **length 0**).

Example:

- For the alphabet $\Sigma = \{0, 1\}$, the set of strings in Σ^* is:
 - $\{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \dots\}$
- **Subtlety:** 0 as a **string**, and 0 as an **alphabet** look the **same**.
 - **Context** determines the type.

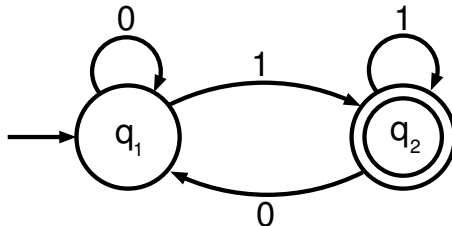
- A **language** is a subset of Σ^* for some alphabet Σ .
- **Example 1:** The set of strings over $\{0,1\}$ with **no two** consecutive 1's.
 - $\{\epsilon, 0, 1, 00, 01, 10, 000, 001, 010, 100, 101, 0000, 0001, 0010, 0100, 0101, 1000, 1001, 1010, \dots\}$
- **Example 2:** The set of strings over $\{a,b,c\}$ with **unequal** number of a's and b's.
 - $\{a, b, aa, ac, bb, bc, ca, cb, aaa, bbb, aab, baa, aba, aac, caa, aca, bbc, cbb, bcb, bba, abb, bab, cca, ccb, cac, cbc, bcc, acc, \dots\}$

Finite Automata

- Informally, finite automata are finite collections of **states** with **transition rules** for going from one state to another.
- There is a **start** state and (one or more) **accept** states.

Representation: Simplest representation is often a graph.

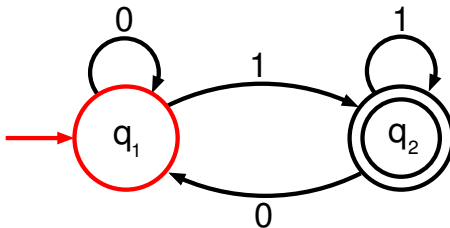
- Nodes denote states, and arcs indicate state transitions.
- Labels on arcs denote the **cause** of transition.



- The above automaton only accepts binary strings **ending in 1**.

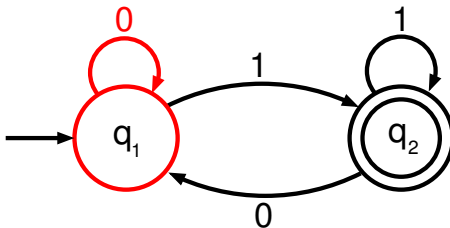
An example

- 01100011100101



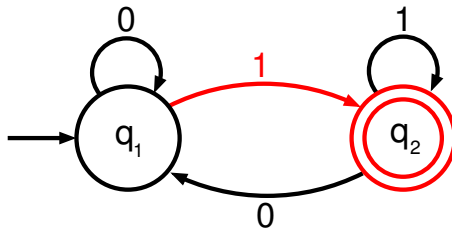
An example

- 01100011100101



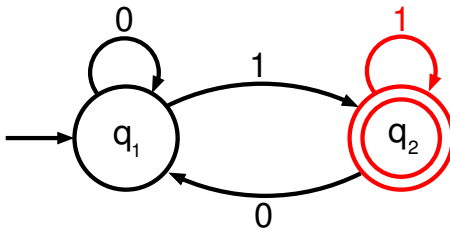
An example

- 01100011100101



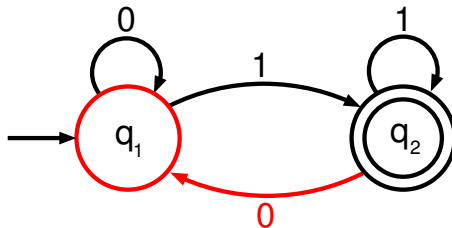
An example

- 01100011100101



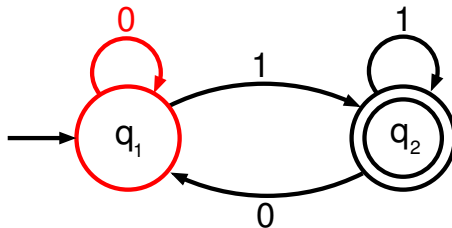
An example

- 01100011100101



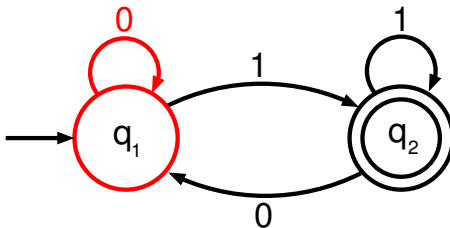
An example

- 01100011100101



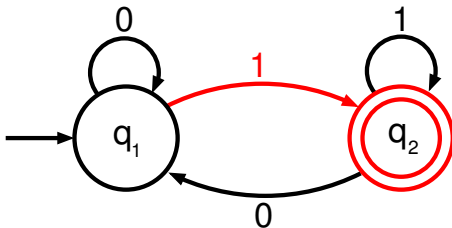
An example

- 01100011100101



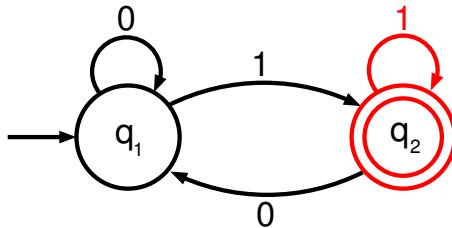
An example

- 01100011100101



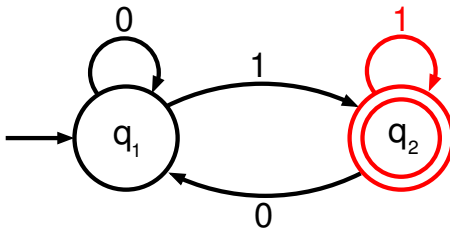
An example

- 01100011100101



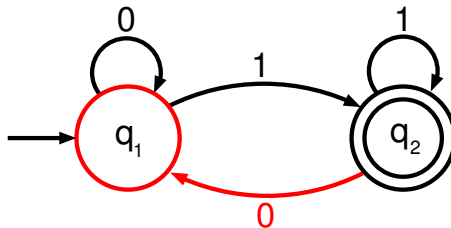
An example

- 01100011100101



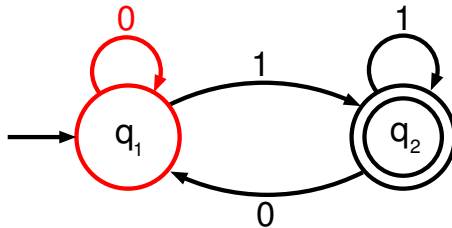
An example

- 01100011100101



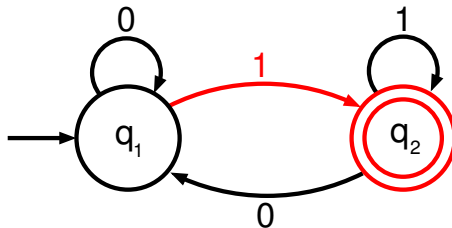
An example

- 01100011100101



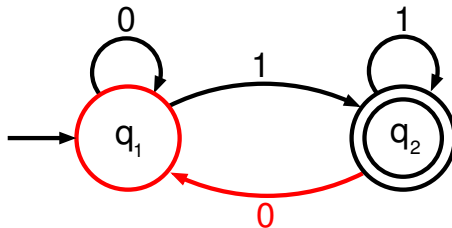
An example

- 01100011100101



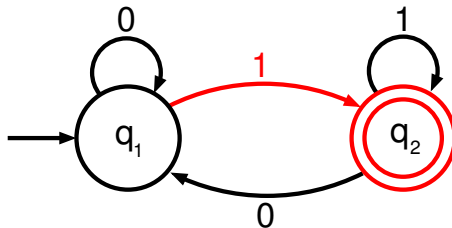
An example

- 01100011100101



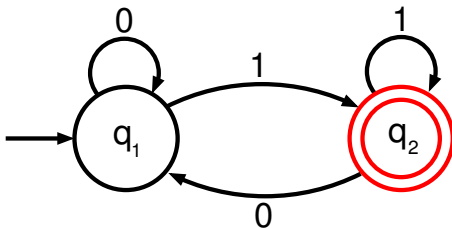
An example

- 01100011100101



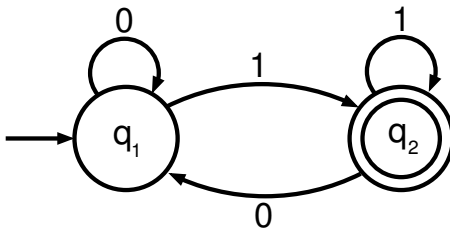
An example

- Accept!



An example

- **Exercise:** Verify that 000111010110 is rejected.



Definition

A DFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ consisting of:

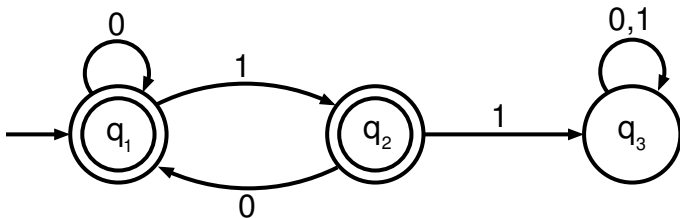
- A finite set of states Q ,
- A set of input alphabets Σ ,
- A transition function $\delta : Q \times \Sigma \rightarrow Q$,
- A start state q_0 , and
- A set of accept states $F \subseteq Q$.

The transition function δ :

- Takes two arguments, a state q and an alphabet a .
- $\delta(q, a)$ = the state the DFA goes to when it is in state q and the alphabet a is received.

Graph representation of DFA's

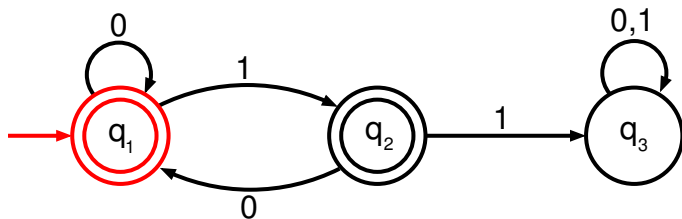
- Nodes correspond to states.
- Arcs represent transition function.
 - Arc from state p to state q labeled by all those input symbols that have transitions from p to q .
- Incoming arrow from outside denotes start state.
- Accept states indicated by double circles.



- Accepts all binary strings **without** two consecutive 1's.

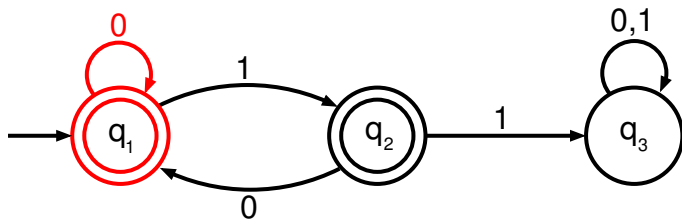
Example

- 011



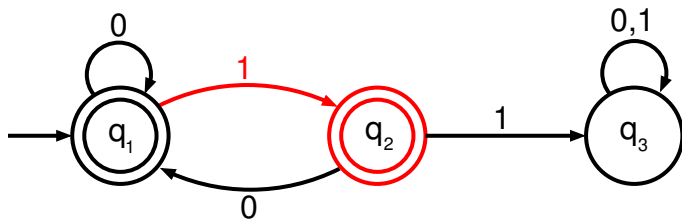
Example

- 011



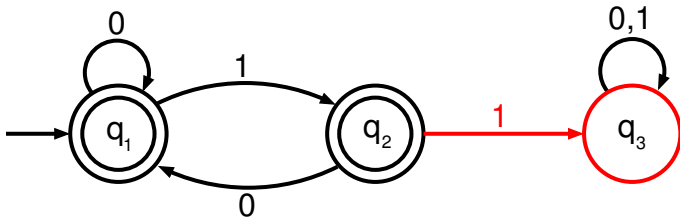
Example

- 011



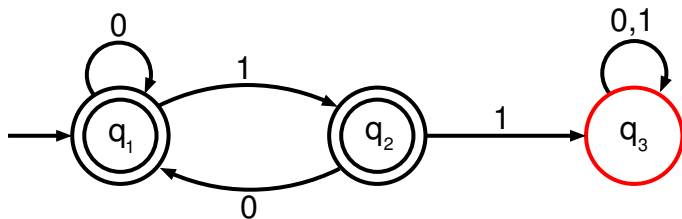
Example

- 011



Example

- 011

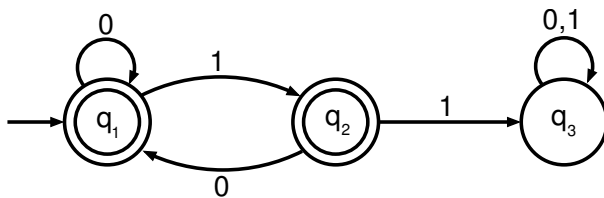


Transition table for DFA's

→

	0	1
q_1^*	q_1	q_2
q_2^*	q_1	q_3
q_3	q_3	q_3

- A row for each **state**, a column for each **alphabet**.
- Accept states are **starred**.
- Arrow for the start state.



Extended transition function

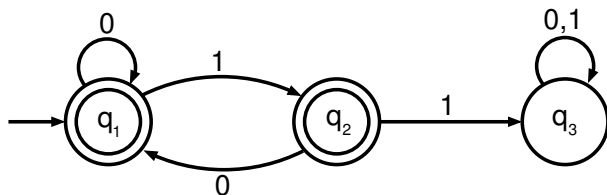
- Effect of a **string** on a DFA can be described by **extending** δ to a state and a string.
- Induction on **length** of the input string.
- **Basis:** $\delta(\mathbf{q}, \epsilon) = \mathbf{q}$
- **Induction:** $\delta(\mathbf{q}, \mathbf{wa}) = \delta(\delta(\mathbf{q}, \mathbf{w}), \mathbf{a})$
 - \mathbf{w} is a **string**, \mathbf{a} is an input **alphabet**.
 - **Convention:** $\mathbf{w}, \mathbf{x}, \mathbf{y}, \dots$ are strings, $\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$ are alphabets.
- For a DFA, extended δ is computed for state \mathbf{q} and inputs $\mathbf{a}_1 \mathbf{a}_2 \dots \mathbf{a}_n$ by following a path starting at \mathbf{q} and selecting arcs with labels $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$ in turn.

Example using transition table

→

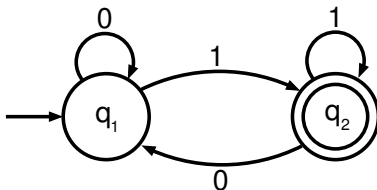
	0	1
q_1^*	q_1	q_2
q_2^*	q_1	q_3
q_3	q_3	q_3

$$\begin{aligned}\delta(q_2, 011) &= \delta(\delta(q_2, 01), 1) = \delta(\delta(\delta(q_2, 0), 1), 1), 1) \\ &= \delta(\delta(q_1, 1), 1) = \delta(q_2, 1) = q_3\end{aligned}$$



Language of a DFA

- Automata of all kinds define languages.
- If A is an automaton, $L(A)$ is its language.
- For a DFA A , $L(A)$ is the set of strings labeling paths from the start state to an accept state.
- Formally, $L(A) =$ set of strings w such that $\delta(q_0, w)$ is in F .
- **Example:** 01100011100101 is in $L(A)$, where A is:

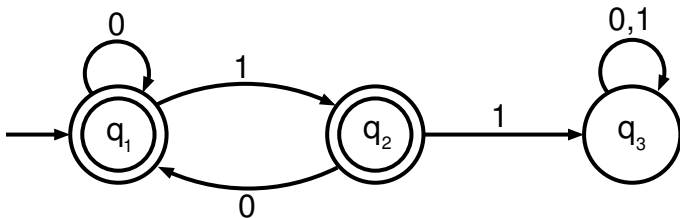


Proofs of set equivalence

Question

How to prove that two different sets are in fact the **same**?

- Here, **T** is the set of strings of **0**'s and **1**'s with no **consecutive 1**'s and $L = \{w \mid M \text{ accepts } w\}$, where **M** is:

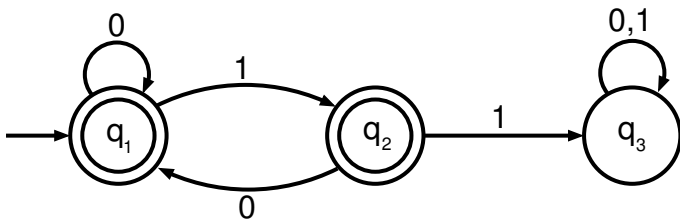


Proofs of set equivalence

Question

How to prove that two different sets are in fact the **same**?

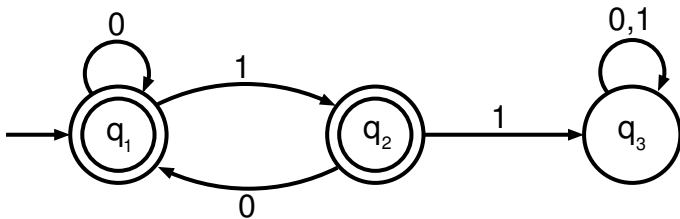
- In general, to show $T = L$, we need to prove two parts:
 - If w is in L , then w is in T , i.e., $L \subseteq T$.
 - If w is in T , then w is in L , i.e., $T \subseteq L$.



- **To prove:** If $w \in L$, then w has no consecutive 1 's.
- Proof is **induction** on length of w .
- **Important trick:** Expand the inductive hypothesis to be more detailed than you need.

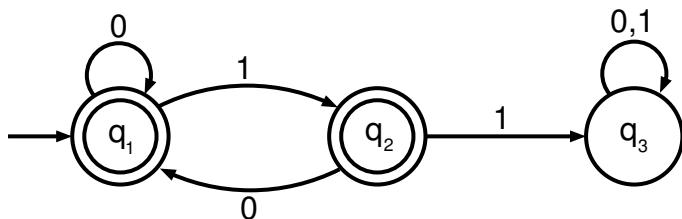
Part 1: $L \subseteq T$ (Inductive Hypothesis)

- ① If $\delta(q_1, w) = q_1$, then w has no consecutive 1's and does not end in 1.
- ② If $\delta(q_1, w) = q_2$, then w has no consecutive 1's and ends in a single 1.
- **Basis:** $|w| = 0$, i.e., $w = \varepsilon$.
 - ① holds as ε has no 1's at all.
 - ② holds *vacuously*, since $\delta(q_1, \varepsilon)$ is not q_2 .
- **Note:** $p \Rightarrow q$ is *always true*, if p is *false*.



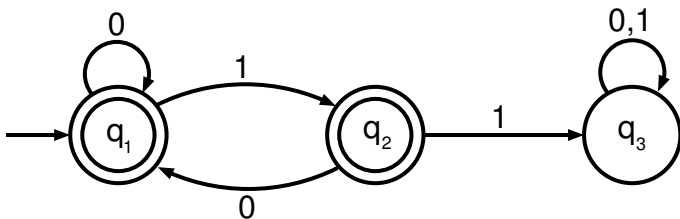
Part 1: $L \subseteq T$

- 1 If $\delta(q_1, w) = q_1$, then w has no consecutive 1's and does not end in 1.
 - 2 If $\delta(q_1, w) = q_2$, then w has no consecutive 1's and ends in a single 1.
- **Inductive step:** Assume **IH** is true for strings shorter than w , where $|w| \geq 1$.
 - Because w is not empty, we can write $w = xa$, where a is the last alphabet of w and x is the string that precedes.
 - **IH** is true for x .



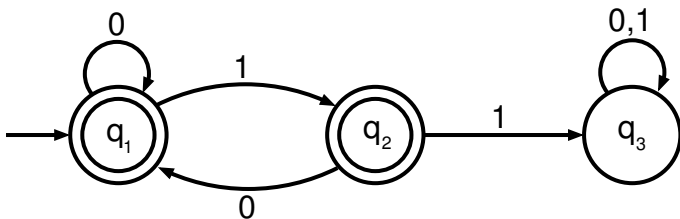
Part 1: $L \subseteq T$

- ① If $\delta(q_1, w) = q_1$, then w has no consecutive 1's and does not end in 1.
- ② If $\delta(q_1, w) = q_2$, then w has no consecutive 1's and ends in a single 1.
- **Inductive step:** Assume **IH** is true for strings shorter than w , where $|w| \geq 1$.
 - ① for w is $\delta(q_1, w) = q_1$.
- Since $\delta(q_1, w) = q_1$, $\delta(q_1, x)$ must be q_1 or q_2 and a must be 0.



Part 1: $L \subseteq T$

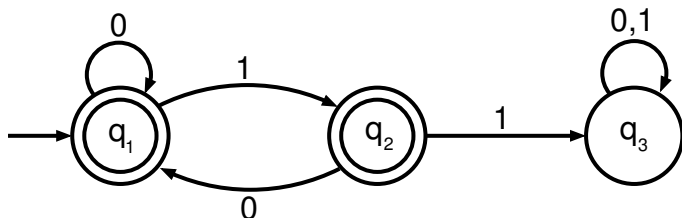
- 1 If $\delta(q_1, w) = q_1$, then w has no consecutive 1's and does not end in 1.
 - 2 If $\delta(q_1, w) = q_2$, then w has no consecutive 1's and ends in a single 1.
- **Inductive step:** Assume **IH** is true for strings shorter than w , where $|w| \geq 1$.
 - 2 for w is $\delta(q_1, w) = q_2$.
 - Since $\delta(q_1, w) = q_2$, $\delta(q_1, x)$ must be q_1 and a must be 1.



- **To prove:** If w has no consecutive 1 's, then $w \in L$.
- We prove the **contrapositive**.
- **Note:** $p \Rightarrow q$ is equivalent to $\neg q \Rightarrow \neg p$.
- **To prove (contrapositive):** If $w \notin L$, then w has no 11 's.

Part 2: $T \subseteq L$

- Simple **induction** on length of w .
- The only way w is not accepted is if it gets to q_3 .
- The only way to get to q_3 is if $w = x1y$.
- If $\delta(q_1, x) = q_2$, then surely $x = z1$ (from Part 1: $L \subseteq T$).
- $\Rightarrow w = z11y!$



Definition

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ **accepts** w if there exists a sequence of states r_0, r_1, \dots, r_n in Q with three conditions:

- $r_0 = q_0$
- $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, \dots, n - 1$, and
- $r_n \in F$

- Condition 1 says that M starts in the start state q_0 .
- Condition 2 says that M follows δ between two states.
- Condition 3 says that last state is an **accept** state.
- We say that M **recognizes** L if $L = \{w \mid M \text{ accepts } w\}$.

Regular languages

- A language L is **regular** if it is the language of some DFA.
 - **Note:** the DFA must accept **only** the strings in L .
- Some languages are not regular (more later).
 - Intuitively, DFA's are **memoryless**.

A regular language

Example

Let $L = \{w \mid w \in \{0,1\}^* \text{ and } w, \text{ viewed as a binary integer, is divisible by } 5.\}$

