	To ***prove*** that this is true…	If you ***assume*** this is true…
$\forall x.\ A$	Have the reader pick an arbitrary x. We then prove A is true for that choice of x.	Initially, ***do nothing***. Once you find a z through other means, you can state it has property A.
$\exists x.\ A$	Find an x where A is true. Then prove that A is true for that specific choice of x.	Introduce a variable x into your proof that has property A.
$A \to B$	Assume A is true, then prove B is true.	Initially, ***do nothing***. Once you know A is true, you can conclude B is also true.
$A \land B$	Prove A. Then prove B.	Assume A. Then assume B.
$A \lor B$	Either prove $\neg A \to B$ or prove $\neg B \to A$. *(Why does this work?)*	Consider two cases. Case 1: A is true. Case 2: B is true.
$A \leftrightarrow B$	Prove $A \to B$ and $B \to A$.	Assume $A \to B$ and $B \to A$.
$\neg A$	Simplify the negation, then consult this table on the result.	Simplify the negation, then consult this table on the result.

Lecture 06: Functions

- $f : A \to B$ notates that f is a function with **domain** A and **codomain** B
- Rules of functions:
    - **$\forall a \in A.\ \exists b \in B.\ f(a) = b$**
        - f can only be applied to elements of its domain
        - For any x in the domain, $f(x)$ is an element of the codomain
    - **$\forall a_1 \in A.\ \forall a_2 \in A.\ (a_1 = a_2 \to f(a_1) = f(a_2))$**
        - Equal inputs produce equal outputs
- Piecewise functions
    - For every x in the domain, at least one rule has to apply
    - All applicable rules should give the same result
    - Example:

$$f(n) = \begin{cases} k & \text{if } \exists k \in \mathbb{N}.\, n = 2k \\ -(k+1) & \text{if } \exists k \in \mathbb{N}.\, n = 2k+1 \end{cases}$$

Lecture 06: Special types of functions

$f : A \to B$ is **injective** (one-to-one) if either of these equivalent statements is true:

$$\forall x_1 \in A.\ \forall x_2 \in A.\ (x_1 \neq x_2 \to f(x_1) \neq f(x_2))$$

$$\forall x_1 \in A.\ \forall x_2 \in A.\ (f(x_1) = f(x_2) \to x_1 = x_2)$$

An injective function associates at most one element of the domain with each element of the codomain.

$f : A \to B$ is **surjective** (onto) if it has this property:

$$\forall b \in B.\ \exists a \in A.\ f(a) = b$$

A surjective function associates at least one element of the domain with each element of the codomain.

$f : A \to B$ is **bijective** if it is both injective and surjective.

Lecture 07: Function Composition

If we have two functions $f : A \to B$ and $g : B \to C$, the **composition of f and g, denoted $g \circ f$**, is a function where

- $g \circ f : A \to C$
- $(g \circ f)(x) = g(f(x))$
  - Notice the parentheses around $(g \circ f)$

Lecture 08: Set Theory Revisited

	Is defined as…	To ***prove*** that this is true…	If you ***assume*** this is true…
$S \subseteq T$	$\forall x \in S.\ x \in T$	Pick an arbitrary $x \in S$. Prove $x \in T$	Initially, ***do nothing***. Once you find some $x \in S$, conclude $x \in T$.
$S = T$	$S \subseteq T \land T \subseteq S$	Prove $S \subseteq T$. Then prove $T \subseteq S$.	Assume $S \subseteq T$ and $T \subseteq S$.
$x \in A \cap B$	$x \in A \land x \in B$	Prove $x \in A$. Then prove $x \in B$.	Assume $x \in A$. Then assume $x \in B$.
$x \in A \cup B$	$x \in A \lor x \in B$	Either prove $x \in A$ or prove $x \in B$.	Consider two cases: Case 1: $x \in A$. Case 2: $x \in B$.
$X \in \wp(A)$	$X \subseteq A$.	Prove $X \subseteq A$.	Assume $X \subseteq A$.
$x \in \{ y \mid P(y) \}$	$P(x)$	Prove $P(x)$.	Assume $P(x)$.

Lecture 09: Graphs

- An **undirected graph** is an ordered pair G = (V, E), where
  - Vis a set of nodes, which can be anything, and
  - E is a set of edges, which are unordered pairs of nodes drawn from V.
  - Because an unordered pair is a set {a, b} of two elements a ≠ b, self-loops (edges from nodes to themselves) are not allowed.
- A **directed graph** (or digraph) is an ordered pair G = (V, E), where
  - V is a set of nodes, which can be anything, and
  - E is a set of edges, which are ordered pairs of nodes drawn from V.
- A **vertex cover** of an undirected graph G = (V, E) is a set C ⊆ V such that:

$$\forall x \in V.\ \forall y \in V.\ (\{x, y\} \in E \to (x \in C \lor y \in C))$$

("Every edge has at least one endpoint in C.")

- An **independent set** in an undirected graph G = (V, E) is a set I ⊆ V such that:

$$\forall u \in I.\ \forall v \in I.\ \{u, v\} \notin E.$$

("No two nodes in I are adjacent.")

Lecture 10: Graph Traversals

Given a graph G = (V, E):

- Two nodes u, v ∈ V are **adjacent** if we have {u, v} ∈ E.
- A **walk** is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.
  - The length of a walk of n nodes is n–1.
- A **closed walk** is a walk from a node back to itself.
  - (By convention, a closed walk cannot have length zero.)
- A **path** is a walk that does not repeat any nodes.
- A **cycle** is a closed walk that does not repeat any nodes or edges except the first/last node.
- A node v is **reachable** from a node u if there is a path from u to v.
- G is called **connected** if all pairs of distinct nodes in G are reachable.
- A **connected component** (or CC) of G is a set consisting of a node and every node reachable from it.
- The **degree** of a node v is the number of nodes that v is adjacent to.

Lecture 11: Pigeonhole Principle

The Pigeonhole Principle:

If m objects are distributed into n bins and m > n, then at least one bin will contain at least two objects.

The Generalized Pigeonhole Principle:

If m objects are distributed into n bins, then

- some bin will have at least $\lceil m/n \rceil$ objects in it, and
- some bin will have at most $\lfloor m/n \rfloor$ objects in it.

Theorem on Friends and Strangers: Color each edge of K_6 red or blue. The resulting graph contains a monochrome copy of K_3.

Lecture 12: Induction

A **proof by induction** is a way to show that some result P(n) is true for all natural numbers n. In a proof by induction, there are three steps:

- Prove that P(0) is true. This is called the basis or the base case.
- Prove that if P(k) is true, then P(k+1) is true.
  - This is called the inductive step.
  - The assumption that P(k) is true is called the inductive hypothesis.
- Conclude, by induction, that P(n) is true for all n $\in \mathbb{N}$.

Lecture 13: Induction Variants

Induction starting at m:

- Prove that P(**m**) is true. This is called the basis or the base case.
- Prove that if P(k) is true, then P(k+1) is true.
  - This is called the inductive step.
  - The assumption that P(k) is true is called the inductive hypothesis.
- Conclude, by induction, that P(n) is true for all natural numbers ≥ **m**

Induction can also be taken with **bigger step sizes** (prove if P(k) is true, then P(k + x) is true for some x) or **multiple base cases**.

Lecture 13: Inducting Up and Down

Building up: If the **predicate P(n)** is existentially quantified, start with the object provided by assuming that P(k) is true.

Building down: If the **predicate P(n)** is universally quantified, start by picking an arbitrary object needed to show P(k + 1) is true.

Lecture 13: Complete Induction

- Define some predicate $P(n)$ to prove by induction on n.
- Choose and prove a base case (probably, but not always, $P(0)$).
- Pick an arbitrary $k \in \mathbb{N}$ and assume that **$P(0)$, $P(1)$, $P(2)$, …, and $P(k)$ are all true.**
  - This is the only difference between complete induction and normal induction.
  - You can use complete induction when you need a stronger assumption during your inductive step.
  - If your base case was not $P(0)$, you would start from whatever your base case(s) were up to k, instead of starting at 0 in this assumption.
- Prove $P(k+1)$.
- Conclude that $P(n)$ holds for all $n \in \mathbb{N}$.

Lecture 14: Languages

- An **alphabet** is a finite, nonempty set of symbols called characters. Typically, we use the symbol Σ.
- A **string** over Σ is a *finite* sequence of characters drawn from Σ.
- The **empty string** has no characters and is denoted ε.
- L is a **language over Σ** if it is a set of strings over Σ.
- The **set of all strings** composed from letters in Σ is denoted Σ^*.
- **Concatenation:**
    - Of strings: If $w \in \Sigma^*$ and $x \in \Sigma^*$, wx is the string formed by putting all the characters of x onto the end of w.
    - Of languages: $L_1L_2 = \{ x \mid \exists w_1 \in L_1.\ \exists w_2 \in L_2.\ x = w_1w_2 \}$
- **Language powers:** Defined recursively. $L^0 = \{\varepsilon\}$ and $L^{n+1} = LL^n$
- **Kleene closure:** $L^* = \{ w \in \Sigma^* \mid \exists n \in \mathbb{N}.\ w \in L^n \}$

Lecture 14: Finite Automata

- A collection of **states** linked by **transitions**
- One state is the **start state**. The computation begins in that state.
- The computation proceeds from left to right over the **input string**. When the automaton sees a character, it follows the transition with that label.
- Some states are **accepting states** (double ring). If the device ends in an accepting state after seeing all the input, it accepts the input. Otherwise, it rejects.

Lecture 14: DFA Rules

- DFAs are defined relative to some alphabet Σ.
- For each state in the DFA, there must be **exactly one** transition defined for each symbol in Σ.
- There is a unique start state.
- There are zero or more accepting states.

The language of D, denoted $\mathscr{L}(D)$, is $\{ w \in \Sigma^* \mid D \text{ accepts } w \}$

Lecture 15: NFA Rules

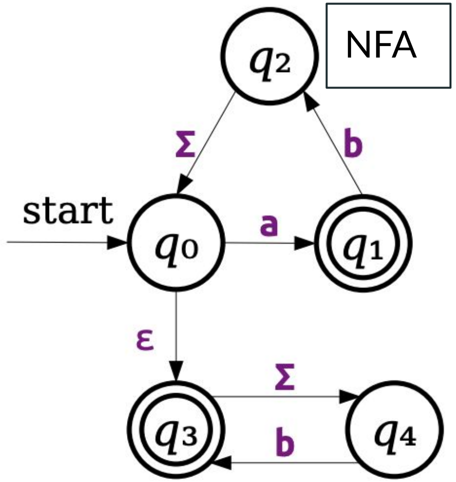
- NFAs have no restrictions on how many transitions are allowed per state.
- NFAs can use ε-transitions, which can be taken at any time.
- An NFA accepts a string if there is **some** sequence of choices that leads to an accepting state after every character from the string has been read.
- Two intuitions: “Perfect guessing”, “massive parallelism”

Lecture 16: The Subset Construction

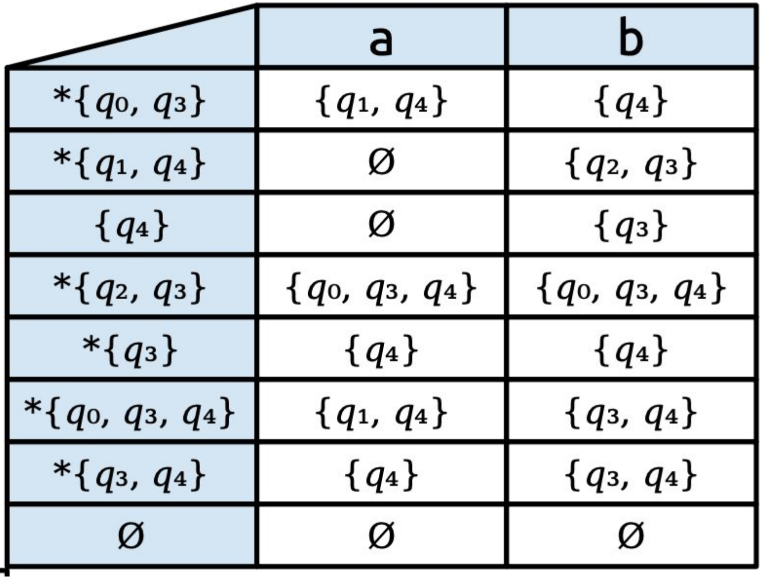
Process to convert an NFA into a DFA: Guide to the Subset Construction

Make a table of all the simultaneous states you can occupy, and the path options from each set of states

At most, the DFA will have $2^{|S|}$ states for an NFA with S states because $|\wp(S)| = 2^{|S|}$



Tabular DFA



	a	b
$*\{q_0, q_3\}$	$\{q_1, q_4\}$	$\{q_4\}$
$*\{q_1, q_4\}$	$\varnothing$	$\{q_2, q_3\}$
$\{q_4\}$	$\varnothing$	$\{q_3\}$
$*\{q_2, q_3\}$	$\{q_0, q_3, q_4\}$	$\{q_0, q_3, q_4\}$
$*\{q_3\}$	$\{q_4\}$	$\{q_4\}$
$*\{q_0, q_3, q_4\}$	$\{q_1, q_4\}$	$\{q_3, q_4\}$
$*\{q_3, q_4\}$	$\{q_4\}$	$\{q_3, q_4\}$
$\varnothing$	$\varnothing$	$\varnothing$

Tournament of Victory Chains: M10 P4

Let T be a tournament. A ***victory chain*** in T is a way of listing the $n \geq 0$ players of T such that each player in the list, except for the very last, won her game against the next person in the list.

Prove the following surprising fact by induction: every tournament has at least one victory chain.

Recommended Practice Problems

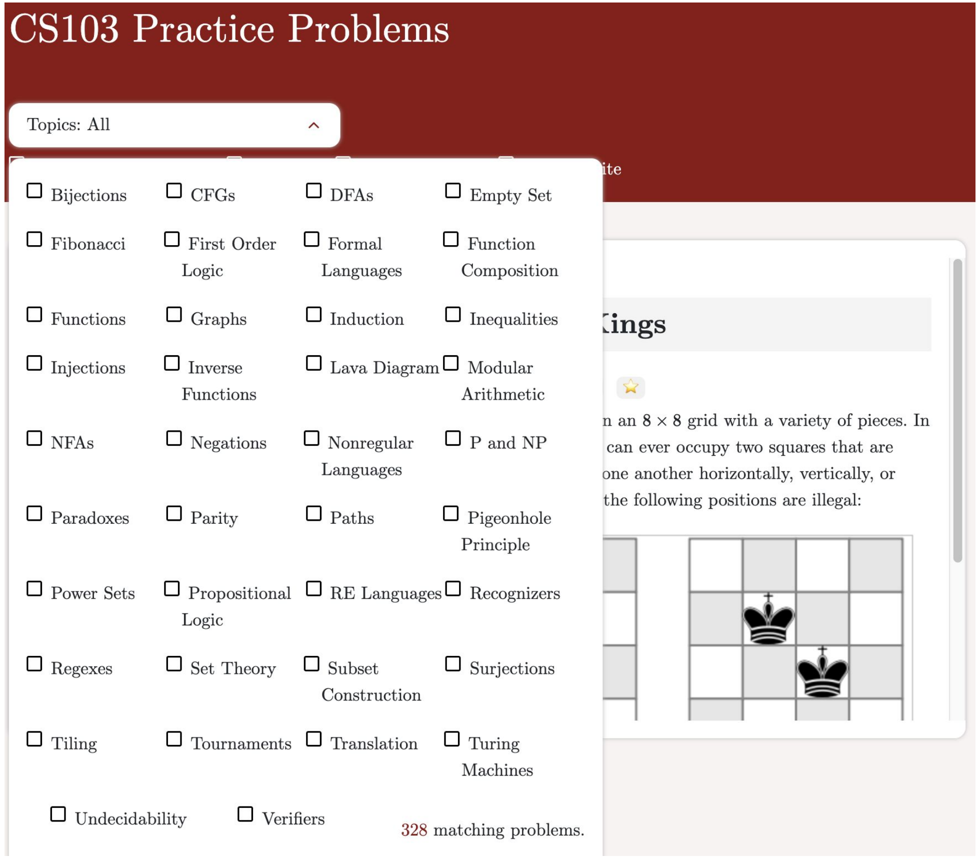
On Printable Practice Test:

- Botanical Graphs - Graphs
- Paying the Troll Toll - Induction
- Exploring a Function - Functions

On other practice exams:

- Powered Power Sets - Set Proofs
- Transitive Sets - Set Proofs
- Bipartite Complements - Graphs / Pigeonhole
- Perfection from Imperfection - Functions
- Cycle-Free Tournaments - Challenging: Graphs, Pigeonhole, & Induction
- Stable Triangles - Induction, Graphs

More Practice Problems



CS103 Practice Problems

Topics: All

☐ Bijections	☐ CFGs	☐ DFAs	☐ Empty Set
☐ Fibonacci	☐ First Order Logic	☐ Formal Languages	☐ Function Composition
☐ Functions	☐ Graphs	☐ Induction	☐ Inequalities
☐ Injections	☐ Inverse Functions	☐ Lava Diagram	☐ Modular Arithmetic
☐ NFAs	☐ Negations	☐ Nonregular Languages	☐ P and NP
☐ Paradoxes	☐ Parity	☐ Paths	☐ Pigeonhole Principle
☐ Power Sets	☐ Propositional Logic	☐ RE Languages	☐ Recognizers
☐ Regexes	☐ Set Theory	☐ Subset Construction	☐ Surjections
☐ Tiling	☐ Tournaments	☐ Translation	☐ Turing Machines
☐ Undecidability	☐ Verifiers		

328 matching problems.

ite

ings

n an 8×8 grid with a variety of pieces. In
can ever occupy two squares that are
one another horizontally, vertically, or
the following positions are illegal: