A Warning

• Most of this should be review

• I'm going to breeze through the basics
  – Spend a little more time on the weird variants

• Come to Office Hours (or Discussion) if you need more review
Topic 01: Review

• Propositions, Predicates, and Operators
• Quantifiers
• Sets
• Proofs (focusing on Induction)
• Trees
Propositions

- Proposition:

- Simple Proposition:

- Compound Proposition:
Propositions

- **Proposition:**
  - A claim that is either true or false with respect to an associated context

- **Simple Proposition:**
  - A proposition with no logical operators

- **Compound Proposition:**
  - A proposition containing one or more logical operators
Propositions and Math

- All equalities and inequalities are propositions

\[ a = b \]
\[ a < c \]
\[ 0 \leq d \leq e \]
Predicates

- Predicate
Predicates

• Predicate
  - A statement that includes at least one variable and will evaluate to either true or false when the variable(s) are assigned values.

• In other words:
  - A function that returns a boolean
Predicates

\[ H(d) = "\text{on date } d \text{ it was hot in Tucson}" \]
Predicates

\[ H(d) = "\text{on date } d \text{ it was hot in Tucson}" \]

\[ H(1 \text{ Aug}) = \text{true} \]
\[ H(1 \text{ Jan}) = \text{false} \]
Multivariable Predicates

\[ P(a, b) = "a is a parent of b" \]

\[ P(Russ, Eric) = ? \]
\[ P(Emily, Eric) = ? \]
\[ P(Rose, Eric) = ? \]
Multivariable Predicates

$P(a, b) =$ "a is a parent of b"

$P(Russ, Eric) = true$
$P(Emily, Eric) = true$
$P(Rose, Eric) = false$
Logical Operators

- Conjunction (AND)
- Disjunction (inclusive OR)
- Exclusive OR (XOR)
- NOT
Logical Operators

- **Conjunction (AND)**
  - True if both of the inputs are true

- **Disjunction (inclusive OR)**
  - True if either (or both) of the inputs are true

- **Exclusive OR (XOR)**
  - True if one **(but not both)** of the inputs are true

- **NOT**
  - True if the input is false
Logical Operators

- Conjunction (AND)
  
  \[ p \land q \]

- Disjunction (inclusive OR)
  
  \[ p \lor q \]

- Exclusive OR (XOR)
  
  \[ p \oplus q \]

- NOT
  
  \[ \neg p \]  
  \[ \overline{p} \]  
  \[ \neg p \]  

These are the logical operators (for use in proofs).
Logical Operators

- Conjunction (AND)
  \[ p \, && \, q \]

- Disjunction (inclusive OR)
  \[ p \, || \, q \]

- Exclusive OR (XOR)
  \[ p \, != \, q \]

- NOT
  \[ !p \]

These are the C / C++ / Java boolean operators (for use in code).
Topic 01: Review

- Propositions, Predicates, and Operators
- **Quantifiers**
- Sets
- Proofs (focusing on Induction)
- Trees
Quantifiers

- \( \exists x, P(x) \)
- \( \forall x, P(x) \)
Quantifiers

- \( \exists x, P(x) \)
  - There exists at least one value of \( x \) such that \( P(x) \) is true.

- \( \forall x, P(x) \)
  - For all values of \( x \), \( P(x) \) is true.
Quantifiers with Conditions

- $\exists x > 0, P(x)$

- $\exists x \in \mathbb{N}, P(x)$
Quantifiers with Conditions

• $\exists x > 0, P(x)$
  - There exists at least one value of $x$ greater than zero such that $P(x)$ is true.

• $\exists x \in \mathbb{N}, P(x)$
  - There exists at least one natural number $x$ such that $P(x)$ is true.

\[ \exists x \in \mathbb{N}, P(x), x \in \mathbb{N} \]
Quantifications with Multiple Variables

- \( \exists x, y, Q(x, y) \)

- \( \forall x, y, Q(x, y) \)
Quantifications with Multiple Variables

- There exists at least one combination of values of $x$ and $y$ such that $Q(x, y)$ is true.

- For all combinations of $x$ and $y$, $Q(x, y)$ is true.

- For all combinations of $x$ and $y$, $Q(x, y)$ is true.
Quantifications with Multiple Variables

\[ \exists x, y, Q(x, y) = \exists x \exists y Q(x, y) \]

\[ \forall x, y, Q(x, y) = \forall x \forall y Q(x, y) \]
Quantifications with Multiple Variables

- \( \exists x \ \forall y, Q(x, y) \)

- \( \forall x \ \exists y, Q(x, y) \)
Quantifications with Multiple Variables

- \( \exists x \ \forall y, Q(x, y) \)
  - There exists at least value of \( x \) such that, for all values of \( y \), \( Q(x, y) \) is true.

- \( \forall x \ \exists y, Q(x, y) \)
  - For every value of \( x \), there exists at least one value of \( y \) such that \( Q(x, y) \) is true.
Quantifications with Multiple Variables

Class Exercise:
How would we prove these propositions true?

\[ \exists x \ \forall y, \ xy = 0 \]

\[ \forall x \ \exists y, \ x + y = 0 \]
Quantifications with Multiple Variables

\[ \exists x \forall y, xy = 0 \]

**Proof:**
For \( x=0 \), \( xy=0 \) for all \( y \).

\[ \forall x \exists y, x+y = 0 \]

**Proof:**
For all \( x \), we set \( y=-x \), and thus \( x+y=0 \).
Free and Bound Variables

\[ \exists x \forall y, Q(x, y) \]

This is a proposition.

This is a predicate.
Free and Bound Variables

\[ \exists x \forall y, Q(x, y) \]

This is a proposition.

This is also a predicate.

This is a predicate.
Free and Bound Variables

<table>
<thead>
<tr>
<th>Formula</th>
<th>Free Vars</th>
<th>Bound Vars</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q(x, y)$</td>
<td>$x, y$</td>
<td>-</td>
</tr>
<tr>
<td>$\forall y, Q(x, y)$</td>
<td>$x$</td>
<td>$y$</td>
</tr>
<tr>
<td>$\exists x \forall y, Q(x, y)$</td>
<td>-</td>
<td>$x, y$</td>
</tr>
</tbody>
</table>

Bound variables are part of quantifiers.

Free variables are everything else.
Free and Bound Variables

\[ Q'(x) = \forall y, Q(x, y) \]

A predicate in one variable

A predicate in two variables

A predicate in one variable
Topic 01: Review

- Propositions, Predicates, and Operators
- Quantifiers
- Sets
- Proofs (focusing on Induction)
- Trees
Sets

- $x \in S$
- $S \subseteq T$
- $S \subset T$
Sets

- $x \in S$
  - $x$ is an element of set $S$

- $S \subseteq T$
  - $S$ is a subset of $T$

- $S \subset T$
  - $S$ is a proper subset of $T$
Some Formal Definitions

\[ S \subseteq T \equiv \forall \ z \in U ( z \in S \rightarrow z \in T ) \]
Some Formal Definitions

$S \subseteq T \equiv \forall z \in U \left( z \in S \rightarrow z \in T \right)$

“All elements that are in S are also in T”
Some Formal Definitions

\[ S \subseteq T \equiv \forall \ z \in U (z \in S \rightarrow z \in T) \]

Did you notice how implication was useful here?

What happens when \( z \notin S \)?
Some Formal Definitions

$S \subseteq T \equiv \forall z \in U(z \in S \rightarrow z \in T)$

$S \subset T \equiv \forall z \in U(z \in S \rightarrow z \in T) \land \exists z \in U(z \in T \land z \not\in S)$

What is the formal definition of $\subset$?
Some Formal Definitions

\[ S \subseteq T \equiv \forall \ z \in U (z \in S \rightarrow z \in T) \]

\[ S \subset T \equiv \forall \ z \in U (z \in S \rightarrow z \in T) \land \exists z \in U (z \in T \land z \notin S) \]
Some Formal Definitions

\[ S \subseteq T \equiv \forall \ z \in U (z \in S \rightarrow z \in T) \]

\[ S \subset T \equiv \forall \ z \in U (z \in S \rightarrow z \in T) \land \exists \ z \in U (z \in T \land z \notin S) \]

“All elements that are in S are also in T … AND … there exists at least one element in T which is not in S”
• What's the difference between these two things?

\[ \{a, b\} \]

\[(a, b)\]
Sets vs. Tuples

• What's the difference between these two things?

\{a, b\} → This is a set. The order of the elements does not matter.

(a, b) → This is an ordered pair. The order of the elements matters a lot.
Sets vs. Tuples

• What's the difference between these two things?

\{ a, b, c, d \}

\( (a, b, c, d) \)

An “ordered pair” with many elements is called a tuple.
Topic 01: Review

- Propositions, Predicates, and Operators
- Quantifiers
- Sets
- **Proofs** (focusing on Induction)
- Trees
What is a Proof?

- Logical argument which establishes a definite proposition
  - One or more hypotheses
  - Draw a conclusion
What is a Proof?

• What you must do:
  – Give an airtight argument!
  – Be clear (communicate)
  – Don't skip steps (prove to me that you understand this!)
Some Definitions

- **Conjecture** – A statement with an unknown truth value
- **Theorem** – A conjecture that has been shown to be true
- **Proof** – A sound argument which establishes the truth of a theorem

These are **NOT** synonyms!
Some Definitions

- **Conjecture**
  - A statement with an unknown truth value

- **Theorem**
  - A conjecture that has been shown to be true

- **Proof**
  - A sound argument which establishes the truth of a theorem
What is Induction?

- Two parts:
  - Base Case(s): Prove that the conjecture holds for the simplest things
  - Inductive Step: Prove that if the conjecture holds for smaller things, it also holds for larger things

**NOTE:**
We generally don't care about the difference between Strong and Weak Induction.
Induction

(1) Can climb upon the first step.

(2) Can climb from any step to the next step.
Induction
(over integers)

Base Case
The base case is a single number (often 0), or sometimes multiple different numbers.
Induction
(over integers)

**Inductive Hypothesis:**
The conjecture holds for all of these.

**We try to prove that the conjecture holds for this one new value.**

**Inductive Step**
Assuming that the conjecture holds for all of the smaller values, prove that it holds for the next one.
Induction
(over integers)

Therefore:

It holds for all values greater than or equal to the base case.
Layout of an Inductive Proof

- State your conjecture
- Prove the base case
  - Often trivial (but still be clear!)
- Clearly state the inductive theorem
  - Inductive Hypothesis: The conjecture holds for $x$
  - Inductive Conclusion: The conjecture holds for $x+1$
  - Again, normally needs a full proof
- Restate the conjecture with “thus” or “therefore”
Induction \(!=\) Direct Proof

IMPORTANT NOTE:

- If I ask you to “prove” a conjecture, any method is OK.
- If I ask you to prove using *induction*, you must use induction or you will lose lots of points (maybe all)!
  - Even if a direct proof is the easier way!
Proof by Induction

- Let's prove these in class (by induction):
  - \[ \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \]
  - \( \forall \ n \geq 2, \ n^2 > n + 1, \ n \in \mathbb{Z} \)
  - All non-negative multiples of 9 (in base 10) have digits that sum up to a multiple of 9.
  - (hard) All palindromes with even #s of digits (in base 10) are multiples of 11
Topic 01: Review

- Propositions, Predicates, and Operators
- Quantifiers
- Sets
- Proofs (focusing on Induction)
- Trees
Trees

• A Tree is:
  - Recursive data structure
  - Each node points to two or more other nodes
  - Acyclic

• Terminology to know:
  - Root, Leaf, Internal Node
  - Subtree
  - Parent, Child, Sibling
Trees

Recursive:
The subtree in the box is also another tree, with its own root.
Trees

**Acyclic:**
Loops are impossible in a tree.
Parent Pointers?

The simplest tree has only pointers from a parent to its children.
Parent Pointers?

Some trees also have parent pointers going back up.
Occasionally, a tree **only** has parent pointers.
The depth (or height) of a tree is the number of layers.

Some people count the nodes; some count the links.

In this class, we will count the links, so this tree has depth 3.
The most common type of tree is a **binary tree**, but more children is possible.

In a **k-ary tree**, each node has $k$ children.

In this tree, each node has (up to) three children.
Trees

This is the “first child-next sibling” way of storing a tree.

In this picture, actual pointers are represented with **heavy lines**.
In an **binary search tree**, the value of each node is $<$ those of all nodes to its right, and $>$ that of all nodes to its left.

Duplicates are sometimes allowed.
To find a value in a binary search tree, do a recursive search (analogous to binary search in an array).
In this example, we're looking for the value 9.5 (which doesn't exist).
To insert new nodes, search through the tree and add a new leaf.
Trees
Trees
A recursive algorithm that lists all of the nodes may traverse it using an **inorder**, **preorder**, or **postorder** strategy.
Trees

Traversals:

In-Order: 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Pre-Order: 10 6 4 3 5 8 7 9 14 12 11 13 16 15 17
Post-Order: 3 5 4 7 9 8 6 11 13 12 15 17 16 14 10
Topic 01: Review

- Propositions, Predicates, and Operators
- Quantifiers
- Sets
- Proofs (focusing on Induction)
- Trees

Summary