Propositional and Predicate Logic

CS 536: Science of Programming, Fall 2019

8/24: p.2; 8/26: p.10

A. Why

- Reviewing/overviewing logic is necessary because we'll be using it in the course.
- We'll be using predicates to write specifications for programs.
- Predicates and programs have meaning relative to states.

B. Outcomes

- At the end of this lecture, you should
- Be able to prove simple logical equivalences of propositions from a basic set of rules.
- Know the syntax for predicates (including primitive tests and the quantifiers $\forall$ and $\exists$).
- Understand how states for predicates differ from states for propositions.
- Understand how $\vDash$ works for non-quantified predicates (for quantified ones we'll need state updates, which we'll see next time).

C. In Case You Missed Class Monday

- The course webpages are at http://cs.iit.edu/~cs536/. Read it carefully for policies and refer to it often to download lecture notes and homework assignments. Check myIIT → Blackboard for lecture videos. Policy questions: Can you work with others on homework? How can you recover from a low Exam 1 score?
- Review the material: Propositions are built up from proposition variables, proposition constants, and the connectives $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, and $\neg$. Parentheses and precedence and associativity rules. Syntactic equality ($\equiv$ and $\not\equiv$). Truth table semantics. States (σ, τ), satisfiability and validity for propositions ($\vDash$ and $\nvDash$). Tautologies, contradictions, contingencies.

D. Formal Proofs of Truth

- For propositions, in addition to semantic truth based on truth tables, there is also a notion of **provable truth** based on syntactic manipulation of propositions. E.g., "if $\varphi \wedge \psi$ is provable then $\psi \wedge \varphi$ is provable" or "$\psi \wedge \varphi$ follows from $\varphi \wedge \psi$".
- **Notation**: if $\vdash \varphi \wedge \psi$ then $\vdash \psi \wedge \varphi$. The $\vdash$ symbol is a "turnstile" (compare to $\vDash$ for semantic truth) and it's pronounced "can prove" or something similar.
- **Definition**. Given a set of proof rules, two propositions are **provably equivalent** if each follows from the other according to those rules. E.g., $\varphi \wedge \psi$ and $\psi \wedge \varphi$ are provably equivalent.

Propositional Logic Rules[*]

- You don't need to memorize these rules by name, but you should be able to give the name of a rule. For example, "$(p \to q) \wedge (p \to r) \Rightarrow (p \to r)$ is __________". (Answer: transitivity)
- The rules use the $\Leftrightarrow$ symbol to indicate that each side can be used to prove the other ($\vdash$ lhs implies $\vdash$ rhs and $\vdash$ rhs implies $\vdash$ lhs).
- **Logical implication vs logical equivalence**: Analogously to how $(\varphi \leftrightarrow \psi) \Leftrightarrow T)$ lets us know $\varphi \Leftrightarrow \psi$, if we know $(\varphi \to \psi) \Leftrightarrow T)$ then we say that $\varphi \Rightarrow \psi$ (φ logically implies ψ).[†] Within the basic proof rules below, $\Rightarrow$ appears in the transitivity rule, and it's critical there.

Commutativity

- $p \vee q \Leftrightarrow q \vee p$
- $p \wedge q \Leftrightarrow q \wedge p$
- $(p \leftrightarrow q) \Leftrightarrow (q \leftrightarrow p)$

Associativity

- $(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r)$
- $(p \wedge q) \wedge r \Leftrightarrow p \wedge (q \wedge r)$

Distributivity/Factoring

- $(p \vee q) \wedge r \Leftrightarrow (p \wedge r) \vee (q \wedge r)$
- $(p \wedge q) \vee r \Leftrightarrow (p \vee r) \wedge (q \vee r)$

Transitivity [Note: $\Rightarrow$, not $\Leftrightarrow$ here]

- $(p \to q) \wedge (q \to r) \Rightarrow (p \to r)$
- $(p \leftrightarrow q) \wedge (q \leftrightarrow r) \Rightarrow (p \leftrightarrow r)$

Identity:	$p \wedge T \Leftrightarrow p$ and $p \vee F \Leftrightarrow p$
Idempotentcy:	$p \vee p \Leftrightarrow p$ and $p \wedge p \Leftrightarrow p$
Domination:	$p \vee T \Leftrightarrow T$ and $p \wedge F \Leftrightarrow F$
Absurdity:	$(F \to p) \Leftrightarrow T$
Contradiction:	$p \wedge \neg p \Leftrightarrow F$
Excluded middle:	$p \vee \neg p \Leftrightarrow T$
Double negation:	$\neg\neg p \Leftrightarrow p$

DeMorgan's Laws

- $\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$
- $\neg(p \vee q) \Leftrightarrow (\neg p \wedge \neg q)$

Definition of $\to$ and $\leftrightarrow$

- $(p \to q) \Leftrightarrow (\neg p \vee q)$
- $(p \leftrightarrow q) \Leftrightarrow (p \to q) \wedge (q \to p)$
- *Negation of Comparisons (These are predicate logic)*
- $\neg(e_1 \le e_2) \Leftrightarrow e_1 > e_2$ (similar for $<, >, \ge, =, \ne$)

- *Substitution*: Given φ, ψ, and ξ, let ξ' be the result of substituting a ψ for one or more occurrences of φ inside ξ. If $\varphi \Leftrightarrow \psi$ then $\xi \Leftrightarrow \xi'$. Example: Using $(p \to q) \Leftrightarrow (\neg p \vee q)$ for ξ, p for φ, and $\neg\neg p$ for ψ, by substitution, we know $(\neg\neg p \to q) \Leftrightarrow (\neg p \vee q)$ [8/24]
-

E. *Sample Proofs*

- Proofs in predicate logic give step-by-step reasoning for why we think the truth of one proposition is related to

[*] You don't need to memorize these rules by name, but you should be able to give the name of the rule given its body. I.e., "$(p \to q) \wedge (\psi \to r) \Rightarrow (p \to r)$ is the _______ rule".

[†] Unfortunately, we're running out of word phrases, so "logically implies" in English can mean $\to$ or $\Rightarrow$. But usually you can figure it out from the context (or just write the symbol, which isn't ambiguous).

another.

- Here is a proof of $\neg(\varphi \rightarrow \psi) \Leftrightarrow (\varphi \wedge \neg\psi)$ (also known as "negation of $\rightarrow$").

$\neg(\varphi \rightarrow \psi)$	
$\Leftrightarrow \neg(\neg\varphi \vee \psi)$	Defn $\rightarrow$
$\Leftrightarrow \neg\neg\varphi \wedge \neg\psi$	DeMorgan's Law
$\Leftrightarrow \varphi \wedge \neg\psi$	Double negation

-
- Here is a proof of $((r \rightarrow s) \wedge r) \rightarrow s \Leftrightarrow \text{T}$ ("Modus ponens").

$(r \rightarrow s) \wedge r \rightarrow s$	
$\Leftrightarrow \neg((r \rightarrow s) \wedge r) \vee s$	Defn of $\rightarrow$
$\Leftrightarrow (\neg(r \rightarrow s) \vee \neg r) \vee s$	DeMorgan's Law
$\Leftrightarrow ((r \wedge \neg s) \vee \neg r) \vee s$	Negation of $\rightarrow$ [see above]
$\Leftrightarrow ((r \vee \neg r) \wedge (\neg s \vee \neg r)) \vee s$	Distribute $\vee$ over $\wedge$
$\Leftrightarrow (\text{T} \wedge (\neg s \vee \neg r)) \vee s$	Excluded middle
$\Leftrightarrow (\neg s \vee \neg r) \vee s$	Identity
$\Leftrightarrow \text{T} \vee \neg r$	Excluded middle (see below)
$\Leftrightarrow \text{T}$	Domination

Avoid Unpleasant Levels of Detail

- In the proof above, if we're being picky with details, then the use of excluded middle to go from $(\neg s \vee \neg r) \vee s$ to $\text{T} \vee \neg r$ is

$(\neg s \vee \neg r) \vee s$	
$\Leftrightarrow \neg s \vee (\neg r \vee s)$	Associativity of $\vee$
$\Leftrightarrow \neg s \vee (s \vee \neg r)$	Commutativity of $\vee$
$\Leftrightarrow (\neg s \vee s) \vee \neg r$	Associativity of $\vee$
$\Leftrightarrow \text{T} \vee \neg r$	Excluded middle

- And if we're being even pickier, we should go from $(\neg s \vee s) \vee \neg r$ to $(s \vee \neg s) \vee \neg r$, since the rule for excluded middle says "$\varphi \vee \neg\varphi \Leftrightarrow \text{T}$", not "$\neg\varphi \vee \varphi \Leftrightarrow \text{T}$". To avoid this level of detail, let's agree that associativity and commutativity can be used without mentioning them specifically.
- **Notation**: In propositional logic proofs (and later, predicate logic proofs), we can omit uses of associativity and commutativity rules and treat them as being implicit. (It's still okay to spell them out, of course.)

F. Derived Rules

- If $\varphi \leftrightarrow \psi$ is a tautology (i.e., $(\varphi \leftrightarrow \psi) \Leftrightarrow \text{T}$), then we can use $\varphi \Leftrightarrow \psi$ as a **derived rule**. E.g., to prove modus ponens, we showed $(r \rightarrow s) \wedge r \rightarrow s \Leftrightarrow \text{T}$. Here's an example of using modus ponens. (For r we substitute $\varphi \wedge \psi$; for s we substitute ξ.)

$((\varphi \wedge \psi) \rightarrow \xi) \wedge (\varphi \wedge \psi) \rightarrow \xi$	Modus ponens
$\Leftrightarrow \text{T}$	

- The next proof translates to $p \wedge q \Rightarrow q$, which is sometimes called "and-elimination".

$p \wedge q \to p$	
$\Leftrightarrow \neg(p \wedge q) \vee p$	Defn $\to$
$\Leftrightarrow (\neg p \vee \neg q) \vee p$	DeMorgan's Law
$\Leftrightarrow \mathrm{T} \vee \neg q$	Excluded middle
$\Leftrightarrow \mathrm{T}$	Domination

- Some other common derived rules: [you don't have to memorize these]. Note that or-introduction uses $\Rightarrow$. Similarly as for $\Leftrightarrow$, we can use $\varphi \Rightarrow \psi$ as a derived rule if we have a proof of $(\varphi \to \psi) \Leftrightarrow \mathrm{T}$.
  - *contraposition*: $(p \to q) \Leftrightarrow (\neg q \to \neg p)$
  - *and-introduction*: $p \to (q \to r) \Leftrightarrow p \wedge q \to r$
  - *or-introduction*: $p \Rightarrow p \vee q$
  - *or-elimination*: $(p \to r) \wedge (q \to r) \wedge (p \vee q) \Rightarrow r$
  - *not-introduction*: $(p \to \mathrm{F}) \Leftrightarrow \neg p$

G. Predicate Logic

- In propositional logic, we assert truths about boolean values; in predicate logic, we assert truths about values from one or more "domains of discourse" like the integers.
- We extend propositional logic with **domains** (sets of values), variables whose values range over these domains, and operations on values (e.g. addition). E.g., for the integers we add the set $\mathbb{Z}$, operations $+, -, *, /, \%$ (mod), and relations $=, \neq, <, >, \leq$, and $\geq$. We'll also add arrays of integers and array indexing, as in b[0].
- A **predicate** is a logical assertion that describes some property of values.
- To describe properties involving values, we add basic relations on values (e.g., less-than). We also have rules over these relations, like x * 0 = 0 being a rule of arithmetic.

States for Predicates; Satisfaction and Validity of Predicates (Omitting quantifiers)

- We'll see quantifiers in a bit, but in the meantime, we can still look at the satisfaction relation for predicates.
- **States with bindings for Domain variables**. With propositions, for states, we've been writing sets of bindings like $\{p = \mathrm{T}, q = \mathrm{F}\}$, where proposition variables are bound to boolean values. With predicates, we can also have bindings like x = 0, which bind a domain variable to a domain value. E.g., $\{p = \mathrm{T}, q = \mathrm{F}, \mathrm{x} = 0\}$ is a state now. We might have to give the value a name if we don't know it precisely, for example $\{p = \mathrm{T}, q = \mathrm{F}, \mathrm{x} = \alpha\}$.
- **States as functions**: Technically, a state is a function from variables (i.e., symbols) to values (proposition variables to boolean values, domain variables to domain values).
- **Notation**: $\sigma(\mathrm{x})$ is the value of the state function σ on variable x. (I.e., it's the value α where the binding $\mathrm{x} = \alpha$ appears in σ. $\sigma(e)$ is the value of the expression e given that its variables take their values from σ. Eg., if $\sigma = \{\mathrm{x} = 1, \mathrm{y} = 3\}$, then $\sigma(\mathrm{x}) = 1$, $\sigma(\mathrm{y}) = 3$, and (picking a random expression),

σ(x+y*y) = 10. We'll define the σ(expression) idea more formally later, but for now your intuition should be fine.

- **Notation**: Since σ is a function, it can also be written as a set of ordered pairs. E.g., {x = 1, y = 3} and {(x, 1), (y, 3)} mean the same thing. (I've been using = because it seems more intuitive to people.)
- **Notation**: φ, ψ, ξ, etc. can stand for propositions or predicates. Propositions are also predicates, so technically we don't need to say "propositions or …", but its good to emphasize the distinction.
- **Satisfaction of Unquantified Predicates**. State σ satisfies predicate φ, written $\sigma \vDash \varphi$ is defined as follows.
  - $\sigma \vDash p$ (a proposition variable) if $\sigma(p) = T$.
  - $\sigma \vDash e_1 = e_2$ holds if the values $\sigma(e_1)$ and $\sigma(e_2)$ are equal. Tests <, ≤, >, ≥, and ≠ are similar.
  - $\sigma \vDash \neg\varphi$ holds if $\sigma \nvDash \varphi$ holds.
  - $\sigma \vDash \varphi \wedge \psi$ holds if $\sigma \vDash \varphi$ and $\sigma \vDash \psi$ both hold.
  - $\sigma \vDash \varphi \vee \psi$ holds if $\sigma \vDash \varphi$ or $\sigma \vDash \psi$ or both hold.
  - $\sigma \vDash \varphi \rightarrow \psi$ holds if $\sigma \vDash \neg\varphi \vee \psi$ holds.
  - $\sigma \vDash \varphi \leftrightarrow \psi$ holds if $\sigma \vDash (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$ holds.
- Some points about this definition. First, it seems weird — all we're doing is substituting English words like "and" for logic symbols like "∧". And that's true. All this means is that the proposition connectives behave as you expect them to if you were to write out things in English. Second, the last two rules, for → and ↔, don't break down the φ and ψ into separate parts, they substitute some larger predicate for $\varphi \rightarrow \psi$ and $\varphi \leftrightarrow \psi$. And this is true also; it just means that "→" and "↔" can be derived (i.e., simulated) by the other connectives.

Quantifiers

- When a predicate includes a variable, we have to ask for what values of the variable we think the predicate might be true: Some current value? Every value? Some value?
- We use quantifiers to specify all values, some value, and exactly one value.

Universal Quantification

- A **universally quantified predicate** (or just "**universal**" for short) has the form $(\forall x \in S\,.\,\varphi)$ where S is a set and φ (the body of the universal) is a predicate involving x. E.g., every integer greater than 1 is less than its own square: $(\forall x \in \mathbb{Z}\,.\,x > 1 \rightarrow x < x^2)$[‡].
- Often we leave out the set if it is understood. E.g., $(\forall x\,.\,x > 1 \rightarrow x < x^2)$.

Existential Quantification

- An **existentially quantified predicate** (or just "**existential**" for short) has the form $(\exists x \in S\,.\,\varphi)$ where S is a set and φ (the body of the existential) is a predicate involving x. E.g., there is a nonzero integer that equals its own square: $(\exists x \in \mathbb{Z}\,.\,x \neq 0 \wedge x = x^2)$.

[‡] By the way, the natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$. (In case you've seen an unconventional source that says $0 \notin \mathbb{N}$.)

- Usually the set is understood to be $\mathbb{Z}$ and we leave out. E.g., $(\exists x . x \neq 0 \wedge x = x^2)$.

Syntactic Equality of Quantified Predicates

- For syntactic equality, which variable you quantify over will make a difference for us, so we'll treat: $(\forall x . x > x-1) \not\equiv (\forall y . y > y-1)$. These will be logically equivalent, however: $(\forall x . x > x-1) \Leftrightarrow (\forall y . y > y-1)$.

Bounded Quantifiers — ended here 2019-08-21

- With bounded quantifiers, we abbreviate a quantifier with a condition in the body by moving the condition to the quantifier. We'll take a predicate with bounded quantifier to be ≡ the one without.
  - **Example 1:** $(\forall x > 1 . x < x^2) \equiv (\forall x . x > 1 \rightarrow x < x^2)$. ("For every $x > 1$, $x < x^3$".)
  - **Example 2:** $(\exists x \neq 0 . x = x^2)$. $\equiv (\exists x . x \neq 0 \wedge x = x^2)$. ("There's some nonzero x such that $x = x^3$".)
- **Definition**: A **bounded quantifier** takes the form $Q\varphi . \psi$ as an ≡ abbreviation for $Q\, x\, op\, \xi$ (where Q is $\forall$ and *op* is $\rightarrow$ or Q is $\exists$ and *op* is $\wedge$). More specifically,
  - $\forall\varphi . \psi$ means $\forall x . \varphi \rightarrow \psi$ where x appears in φ and x is understood to be the variable we are quantifying over.
  - $\exists\varphi . \psi$ means $\exists x . \varphi \wedge \psi$ where x appears in φ and x is understood to be the variable we are quantifying over. (Note: it's $\varphi \wedge \psi$ here; compare with $\varphi \rightarrow \psi$ for bounded universals.)
- It's important to use the right connective ($\rightarrow$ for bounded $\forall$ and $\wedge$ for bounded $\exists$). For example, there's a big difference between $(\exists x \in \mathbb{Z} . x > 1 \wedge x = x^2)$ and $(\exists x \in \mathbb{Z} . x > 1 \rightarrow x = x^2)$; the first one is false, the second one is true (any $x \leq 0$ makes the implication true).

Parentheses For Quantified Predicates

- We'll treat $\forall$ and $\exists$ as having low precedence. (Note: Some people use high precedence). So the body of a quantified predicate is as long as possible.
- **Example 3**. $\forall x \in \mathbb{Z} . x > 1 \rightarrow x < x^2$ means $(\forall x \in \mathbb{Z} . ((x > 1) \rightarrow (x < x^2)))$.
- **Example 4**: $\forall x \in \mathbb{Z} . \exists y \in \mathbb{Z} . y \leq x^2$ means $(\forall x \in \mathbb{Z} . (\exists y \in \mathbb{Z} . (y \leq x^2)))$.
- **Notation**: Q means $\forall$ or $\exists$.
- If we have (… $Q\,x$. …) where the two parentheses shown match, then the body can't extend past the right parenthesis, and we get (… $Q\,x$. (…)).
- **Example 5**: $(\exists y \in \mathbb{Z} . y > 0 \wedge x > y) \rightarrow x \geq 1$

  $\equiv ((\exists y \in \mathbb{Z} . ((y > 0) \wedge (x > y))) \rightarrow (x \geq 1))$
- For full parenthesizations, we add parentheses around basic tests, but we still omit them around variables and constants. Let's also omit them around array indexes, so we'll write $(b[x+1] > y)$, not $(b[(x+1)] > y)$. [It's still okay to write $b[(x+1)] > y$ for $(b[(x+1)] > y)$.]
- **Example 6**: $x > 0 \wedge y \leq 0$ expands to $((x > 0) \wedge (y \leq 0))$.

DeMorgan's Laws Extended

- For quantified predicates, there are two more **DeMorgan's Laws**:
  - $(\neg\forall x . \varphi) \Leftrightarrow (\exists x . \neg\varphi)$ and $(\neg\exists x . \varphi) \Leftrightarrow (\forall x . \neg\varphi)$

- With bounded quantifiers, because of how $\rightarrow$, $\neg$, and $\wedge$ are related,
  - $(\neg\forall\varphi.\psi) \Leftrightarrow (\exists\varphi.\neg\psi)$. *I.e.,* $(\neg\forall x.\varphi\rightarrow\psi) \Leftrightarrow (\exists x.\neg(\varphi\rightarrow\psi)) \Leftrightarrow (\exists x.\varphi\wedge\neg\psi) \Leftrightarrow (\exists\varphi.\neg\psi)$.
  - $(\neg\exists\varphi.\psi) \Leftrightarrow (\forall\varphi.\neg\psi)$ *I.e.,* $(\neg\exists x.\varphi\wedge\psi) \Leftrightarrow (\forall x.\neg(\varphi\wedge\psi))$ $\Leftrightarrow (\forall x.\neg\varphi\vee\neg\psi)) \Leftrightarrow (\forall x.\varphi\rightarrow\neg\psi) \Leftrightarrow (\forall\varphi.\neg\psi)$.
- **Example 7**: $\neg(\forall x.x>0) \Leftrightarrow (\exists x.\neg(x>0)) \Leftrightarrow (\exists x.x\leq 0)$
- **Example 8**: $\neg(\forall x>0.x^2=x) \Leftrightarrow (\exists x>0.x^2\neq x)$
- **Example 9**: $\neg(\exists x.x\leq 0\wedge x>0) \Leftrightarrow (\forall x.\neg(x\leq 0\wedge x>0)) \Leftrightarrow (\forall x.x>0\vee x\leq 0))$.

Proofs of Quantified Predicates

- Formal systems for proving predicates are pretty complicated; rather than study one of them, let's rely on an informal idea of how to prove universally and existentially quantified predicates.
- In general, to prove $\forall x.\varphi$, you prove φ but without imposing any restrictions on *x*. If you need to restrict *x*, then this needs to be part of the body of the quantified predicate.
  - **Example 10**: To prove $\forall x\in\mathbb{Z}.x\neq 0\rightarrow x\leq x^2$, we can say "Let x be an integer. Assume that x isn't zero. In that case, $x\leq x^2$."
- To prove $\exists x\in S.\varphi$, you name a **witness value** for *x* and prove φ holds if *x* has that value.
  - **Example 11**: To prove $\exists x\in\mathbb{Z}.x\neq 0\wedge x\geq x^2$, the only value that works as a witness is 1. More generally, there may be multiple witness values that work; we just need to name one.
- If a predicate includes unquantified variables, then for it to be a tautology, it has to hold for all possible values of those quantified variables. It's a contradiction if it fails for all values, and it's a contingency if it holds for some values but not some others. (I.e., unquantified variables are "implicitly universally quantified".)
  - **Example 12**: $x>0\rightarrow\exists y.y^2<x$ is a tautology because $\forall x.(x>0\rightarrow\exists y.y^2<x)$ holds.
  - **Example 13**: $x>0\rightarrow y^2<x$ is a contingency because it holds for some x and y (like x = 2 and y = 1) but fails for others (like $x=y=1$).
  - **Example 14**: $\exists y.(y<0\wedge y>x^2)$ is a contradiction because it fails for every value of x.

Predicate Functions

- Often, we'll give names to predicates and parameterize them. In programming languages, these predicate functions are written as functions that yield a boolean result.
  - **Example 15**: we might define $\texttt{Even}(x)\equiv(x\,\%\,2)=0$, where % is the remainder operator. E.g., $\texttt{Even}(3)\equiv(3\,\%\,2)=0 \Leftrightarrow 1=0 \Leftrightarrow F$.
- In a programming language, the body of a predicate function can be a general program — one that uses loops and decisions. We want our predicates to be simpler than that: We're going to use predicates to augment our programs with specifications, and it won't help if debugging a predicate function body is exactly as hard as debugging a general program.
- So we'll restrict ourselves to predicate functions that take one or more parameter variables and evaluate a predicate on those variables. **Example**: $p(x)\equiv x>y\wedge x<z$.

- The body of the predicate function is (surprise) a predicate that evaluates to true or false, not an expression that yields a number.
  - Function: $\texttt{sqrt}(x) \equiv$ (in pseudocode r where $r = 0$ if $x \le 0$ and where $r * r = x$ if $x \ge 0$.
  - Predicate function: $\texttt{isSqrt}(x, r) \equiv x \le 0 \land r = 0 \lor r * r = x$
- The body of a predicate function can use the parameter variables and the built-in relations for our datatypes (for integers, <, ≤, etc.) along with the propositional connectives (∧, ∨, etc.)
- **Example 16**: Let's define `IsZero(b, m)` to be true if the first `m` elements of `b` are all zero. To help, let's assume `size(b)` gives the number of elements in `b`.
  - Rewriting, `IsZero(b, m)` means that $b[0], b[1], \ldots, b[m-1]$ all $= 0$.
- It might be tempting to write $\texttt{IsZero}(b, m) \equiv b[0] = 0 \land b[1] = 0 \land \ldots \land b[m-1] = 0$
  - But the right hand side is not a predicate; a predicate needs a fixed number of conjuncts being and'ed together.
- To write this as a predicate, we look for a pattern in our informal description: "$b[0], b[1], \ldots, b[m-1]$ all $= 0$" is equivalent to "$b[i] = 0$ for (every) $i = 0, 1, \ldots, m-1$". The implied "every" i tells us we need a universal quantifier ∀.
  - So we can get $\texttt{IsZero}(b, m) \equiv \forall j .\ 0 \le j < m \rightarrow b[j] = 0$. (With bounded quantifiers, $\texttt{IsZero}(b, m) \equiv \forall\ 0 \le j < m .\ b[j] = 0$.
- Another way to look at a description and find an equivalent predicate is to imagine writing a loop to calculate whether the property is true or false.
  - E.g., with "$b[0], b[1], \ldots, b[m-1]$ all $= 0$" we might imagine a loop

```
for i = 0 to m-1
    if b[i] $\neq$ 0 then return false
return true
```

  - The "`for i = 0 to m-1`" tells us we need to search for i in the range $0 \le i < m$.
  - The loop returns true only if **all** the $b[i]$ pass the $= 0$ test; this tells us we need $\forall i$.
  - The general translation for a universal is ∀ *loop var* . ((*var* in search range) → (test on *var*)). For this example, the search range is $0 \le i < m$ (plus an out of bounds test $i < \texttt{size}(b)$) and the test on i is $b[i] = 0$. This gives us $\forall i .\ 0 \le i < m \le \texttt{size}(b) \rightarrow b[i] = 0$.
  - We need a ∃ search if our loop needs to return true as soon as it finds a $b[i]$ that passes the test. E.g., if the property had been "At least one of $b[0], b[1], \ldots, b[m-1] = 0$", we might imagine a loop

```
for i = 0 to m-1
    if b[i] = 0 then return true
return false
```

  - For an existential we translate the loop to ∃ *loop var* . ((*var* in search range) ∧ (test on *var*)). For this example, we get $\exists i .\ 0 \le i < m \le \texttt{size}(b) \land b[i] = 0$.

- **Example 17**: Define `SortedUp(b, m, n)` so that it is true when array `b` is sorted ≤ on the segment `m..n`. As an example, if `b[0..3]` is [1, 3, 5, 2], then `SortedUp(b,0,2)` is true because $1 \le 3$ and $3 \le 5$ but `SortedUp(b,0,3)` is false because we don't have ($1 \le 3$ and $3 \le 5$ and $5 \le 2$).
- Another way to describe `SortedUp(b, m, n)` is that each element in the list `b[m]`, `b[m+1]`, …, `b[n−2]`, `b[n−1]` is ≤ the element to its right.
  - Or expanding further, $b[m] \le b[m+1]$, $b[m+1] \le b[m+2]$, …, $b[n-1] \le b[n]$. We can generalize this to $b[i] \le b[i+1]$ for $i = m, m+1, m+2, \ldots, n-1$. To get a formal predicate, we'll need a ∀ over `i`:
    - $\text{SortedUp}(b, m, n) \equiv \forall i \,.\, 0 \le m \le i < n < \text{size}(b) \to b[i] \le b[i+1]$
    - or, $\text{SortedUp}(b, m, n) \equiv \forall\, 0 \le m \le i < n < \text{size}(b) \,.\, b[i] \le b[i+1]$.
  - We can hoist the parts of this that don't depend on the quantified variable `i`:
    - $\text{SortedUp}(b, m, n) \equiv 0 \le m < n < \text{size}(b) \wedge \forall\, m \le i < n \,.\, b[i] \le b[i+1]$.
- Note: Different generalizations of a property can lead us to different predicates.
- For `SortedUp`, if we generalize ($b[m] \le b[m+1]$, $b[m+1] \le b[m+2]$, …, and $b[n-1] \le b[n]$) to ($b[m+j] \le b[m+j+1]$ for $j = 0, 1, \ldots n-1-m$), we get $\forall\, 0 \le j < n-m \,.\, b[m+j] \le b[m+j+1]$ (and $0 \le m \le n < \text{size}(b)$).
- **Example 18**: Let's find a definition for `Extends(b, b′)` so that it's true if `b′` is an extension of `b`. I.e., $b[0] = b'[0]$, $b[1] = b'[1]$, … for all elements of `b`. Note `b′` can be the same length as `b` or can be longer. E.g., if `b` is [1, 6, 2] and `b′` is [1, 6, 2, 8], then `Extends(b, b′)` is true and `Extends(b′,b)` is false). Here's one solution:
- $\text{Extends}(b, b') \equiv \text{size}(b) \le \text{size}(b') \wedge \forall\, 0 \le k < \text{size}(b) \,.\, b[k] = b'[k]$.)

Propositional and Predicate Logic

CS 536: Science of Programming

A. *Why?*

- Reviewing/overviewing logic is necessary because we'll be using it in the course.
- We'll be using predicates to write specifications for programs.

B. *Objectives*

- At the end of this activity you should be able to:
- Follow and generate simple proofs of propositional formulas, given a set of rules.
- Read and write predicates and logically negate predicates.
- Translate informal descriptions of properties on integers and arrays into formal predicates and predicate functions.

C. *Questions*

1. Fill in the missing rule names in the proof below of $\neg(p \leftrightarrow \psi) \Leftrightarrow (\psi \wedge \neg p) \vee (p \wedge \neg\psi)$, using the rules from the lecture notes. (See page 2.)

$\neg(p \leftrightarrow \psi)$	
$\Leftrightarrow \neg((p \to \psi) \wedge (\psi \to p))$	by defn $\leftrightarrow$
$\Leftrightarrow \neg(p \to \psi) \vee \neg(\psi \to \varphi)$	__________
$\Leftrightarrow ((p \wedge \neg\psi) \vee (\psi \wedge \neg\varphi)$	__________
$\Leftrightarrow (\psi \wedge \neg\varphi) \vee (p \wedge \neg\psi)$	__________

2. Write a formal proof that shows that $(p \to p \vee q)$ (sometimes called the "$\vee$ introduction" rule) is a tautology: Prove $(p \to \varphi \vee q) \Leftrightarrow T$

$p \to p \vee \psi$	
$\Leftrightarrow$ __________	by _________
etc.	

3. Some logical rules can be derived from others. Prove the rule of contraposition by proving $(\varphi \to \psi) \to (\neg\psi \to \neg\varphi) \Leftrightarrow T$ [8/26], using only these rules: Definition of $\to$, double negation, commutativity of $\vee$, and excluded middle. (You may need to use a rule more than once.)

4. Let $q(x, y) \equiv x < y \to y < z \wedge f(x) = 2$. Expand $\neg q(x, y)$ to remove $\neg$ signs: Use the rules to find a predicate equivalent to $\neg(x < y \to y < z \wedge f(x) = 2)$ that doesn't use $\neg$. Hint: Use DeMorgan's laws to move the negation "inward" to smaller and smaller subexpressions. Show your reasoning as a formal proof. (Don't forget the rule names.)

5. What are the minimal and full parenthesizations [8/26] for
   a. $(\forall x . ((\exists y . x > y) \wedge (\exists y . x < y)))$
   b. $\forall x. \neg(\exists y. \varphi \wedge \forall z. \psi)$ [8/26]
   c. $\forall x . \forall y . \exists z . (x \neq y \to x \leq z \wedge z \leq y \vee x > z \wedge z \geq y)$

6. In general, if $\forall x.\forall y.\, P(x, y)$ is valid, is $\forall y.\forall x.\, P(x, y)$ also valid? What about $\exists x.\, \exists y.\, P(x, y)$ and $\exists y.\, \exists x.\, P(x, y)$?
7. Using propositional and predicate proof rules, find a predicate equivalent to $\neg(\forall x.\, \exists y.\, P(x, y))$ that has no negation symbols (i.e., $\neg$), except possibly in front of $P(x, y)$. Write a formal proof that shows each step needed (don't forget the rule names!). Hint: Use DeMorgan's laws to move the negation inward.
8. Repeat the previous question on $\neg(\exists y.\, \forall x.\, P(x, y))$.
9. Write the definition of a predicate function `Repeats(b, m)` that is true exactly when the first `m` elements of `b` match the second `m` elements of `b`: I.e., `b[0] = b[m], b[1] = b[m+1]`, …, `b[m-1] = b[2*m-1]`. (Alternatively, `b[0..m-1]` and `b[m..2*m-1]` are pointwise equal.) Example: If `b` is `[1, 3, 5, 1, 3, 5]`, then `Repeats(b, 3)` is true but `Repeats(b, 2)` is false.

CS 536: Solution to Activity 2 (Propositional and Predicate Logic)

1. DeMorgan's law; Negation of → twice; commutativity of ∨.

2. $p \to p \vee q$

$\Leftrightarrow \neg p \vee (p \vee q)$	Defn →
$\Leftrightarrow (\neg p \vee p) \vee q$	Associativity of ∨
$\Leftrightarrow T \vee q$	Excluded middle
$\Leftrightarrow T$	Domination

3. $(p \to q) \to (\neg q \to \neg p)$

$\Leftrightarrow (\neg p \vee q) \to (\neg q \to \neg p)$	Defn →
$\Leftrightarrow (\neg p \vee q) \to (\neg\neg q \vee \neg p)$	Defn →
$\Leftrightarrow (\neg p \vee q) \to (q \vee \neg p)$	¬¬
$\Leftrightarrow (\neg p \vee q) \to (\neg p \vee q)$	Comm. of ∨
$\Leftrightarrow \neg(\neg p \vee q) \vee (\neg\neg q \vee \neg p)$	Defn →
$\Leftrightarrow T$	Excluded middle (on $(\neg p \vee q)$)

4. If $q(x, y) \equiv x < y \to y < z \wedge f(x) = 2$, then

$\neg q(x, y)$

$\Leftrightarrow \neg(x < y \to y < z \wedge f(x) = 2)$	Defn of q
$\Leftrightarrow x < y \wedge \neg(y < z \wedge f(x) = 2)$	Negation of →
$\Leftrightarrow x < y \wedge (\neg(y < z) \vee \neg(f(x) = 2))$	DeMorgan's Law
$\Leftrightarrow x < y \wedge (y \geq z \vee f(x) \neq 2)$	Negation of comparison, 3 times

5. (Minimal and full parenthesizations)
I've included the outermost parentheses in the full parenthesization.

a. $(\forall x . ((\exists y . x > y) \wedge (\exists y . x < y)))$
Minimal: $\forall x . (\exists y . x > y) \wedge \exists y . x < y$
Full: $(\forall x . ((\exists y . (x > y)) \wedge (\exists y . (x < y))))$

b. $\forall x. \neg(\exists y. p \wedge \forall z. q)$
Minimal: ~~$\forall x. \neg \exists y. p \wedge \forall z. q$~~ [2/18]: $\forall x. \neg(\exists y. p \wedge \forall z. q)$
Full: $(\forall x. (\neg(\exists y. (p \wedge (\forall z. q)))))$

c. $\forall x . \forall y . \exists z . (x \neq y \to x \leq z \wedge z \leq y \vee x > z \wedge z \geq y)$
Minimal: $\forall x . \forall y . \exists z . x \neq y \to x \leq z \wedge z \leq y \vee x > z \wedge z \geq y$
Full: $(\forall x. (\forall y. (\exists z. ((x \neq y) \to (((x \leq z) \wedge (z \leq y)) \vee ((x > z) \wedge (z \geq y)))))))$

6. (Qx.Qy versus Qy.Qx)

a. Yes: $(\forall x.\forall y. P(x, y))$ is valid if and only if $(\forall y.\forall x. P(x, y))$ is valid

b. Yes: $(\exists x.\exists y. P(x, y))$ is valid if and only if $(\exists y.\exists x. P(x, y))$ is valid

7. $\neg(\forall x . \exists y . P(x, y))$

$\Leftrightarrow \exists x . \neg \exists y . P(x, y)$	DeMorgan's Law ¬∀
$\Leftrightarrow \exists x . \forall y . \neg P(x, y)$	DeMorgan's Law ¬∃

8. $\neg(\exists y . \forall x . P(x, y))$
$\Leftrightarrow \forall y . \neg(\forall x . P(x, y))$ DeMorgan's Law $\neg\exists$
$\Leftrightarrow \forall y . \exists x . \neg P(x, y)$ DeMorgan's Law $\neg\forall$

9. Here’s one solution:

$\texttt{Repeats(b, m)} \equiv \forall j.(0 \le j < m \rightarrow m+j < \text{size}(b) \wedge b[j] = b[m+j])$

or $\texttt{Repeats(b, m)} \equiv \forall 0 \le j < m.(m+j < \text{size}(b) \wedge b[j] = b[m+j])$, with bounded quantifiers.

There are other solutions, for example:

$\texttt{Repeats(b, m)} \equiv 0 \le 2*m \le \text{size}(b) \wedge \forall j.(0 \le j < m \rightarrow b[j] = b[m+j])$