UC Davis
ECS20, Winter 2019

Discrete Mathematics for Computer Science

Prof. Raissa D’Souza
(slides adopted from Michael Frank and Haluk Bingöl)

Lecture 3
Foundations of Logic: Overview

• Propositional logic (§1.1-1.2):
  – Basic definitions. (§1.1)
  – Equivalence rules & derivations. (§1.2)

• Predicate logic (§1.3-1.4)
  – Predicates.
  – Quantified predicate expressions.
  – Equivalences & derivations.
Implication Truth Table

- $p \rightarrow q$ is false only when $p$ is true but $q$ is not true.

- $p \rightarrow q$ does not say that $p$ causes $q$!

- $p \rightarrow q$ does not require that $p$ or $q$ are ever true!

- *E.g.* “$(1=0) \rightarrow \text{pigs can fly}”$ is TRUE!

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \rightarrow q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
The contrapositive \( \neg q \rightarrow \neg p \)

- Proving \( p \rightarrow q \equiv \neg q \rightarrow \neg p \) using truth tables:

<table>
<thead>
<tr>
<th></th>
<th>( q )</th>
<th>( \neg q )</th>
<th>( \neg p )</th>
<th>( p \rightarrow q )</th>
<th>( \neg q \rightarrow \neg p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
The cornerstone arguments in logic (Sec 1.6) and Lec 6. Modus Ponens & Tollens

\[
\begin{align*}
\begin{array}{c}
p \\
p \rightarrow q
\end{array} & \quad \text{Rule of \textit{modus ponens}} \\
\therefore q
\end{align*}
\]

\[
\begin{align*}
(p \land (p \rightarrow q)) & \rightarrow q
\end{align*}
\]

“p is sufficient for q”

“the mode of affirming”

\[
\begin{align*}
\begin{array}{c}
\neg q \\
p \rightarrow q
\end{array} & \quad \text{Rule of \textit{modus tollens}} \\
\therefore \neg p
\end{align*}
\]

\[
\begin{align*}
(\neg q \land (p \rightarrow q)) & \rightarrow \neg p
\end{align*}
\]

“q is necessary for p”

“the mode of denying”
Example Equivalence Laws

• **Distributive:** $p \lor (q \land r) \iff (p \lor q) \land (p \lor r)$
  
  $p \land (q \lor r) \iff (p \land q) \lor (p \land r)$

• **De Morgan’s:**
  
  $\neg (p \land q) \iff \neg p \lor \neg q$

  $\neg (p \lor q) \iff \neg p \land \neg q$

• **Trivial tautology/contradiction:**
  
  $p \lor \neg p \iff \text{T}$

  $p \land \neg p \iff \text{F}$
Review: Propositional Logic (§§1.1-1.2)

- Atomic propositions: $p, q, r, \ldots$
- Boolean operators: $\neg \land \lor \oplus \rightarrow \leftrightarrow$
- Compound propositions: $s \equiv (p \land \neg q) \lor r$
- Equivalences: $p \land \neg q \iff \neg (p \rightarrow q)$
- Proving equivalences using:
  - Truth tables.
  - Symbolic derivations. $p \iff q \iff r \ldots$
Predicate Logic
Predicate Logic (§1.3)

- *Predicate logic* is an extension of propositional logic that permits concisely reasoning about whole *classes* of entities.

- Recall algebraic formulas like $x > 3$ are not propositions!

- Predicate logic lets us expand Boolean logic to situations where there is one or more *variables* need to be accounted for.
Propositional Functions

Define a propositional function $P(x)$

• For instance a math expression, $x > 3$
• For some values of $x$, $x > 3$ is $T$
• For other values of $x$, $x > 3$ is $F$
• “$x$” object, “$>3$” predicate

• Or, an English expression,
  $P(x) = “Student \ x \ is \ sleeping”$.

$P$ is called the predicate
$x$ is called the object

A propositional function $P(x)$ takes in an object as an argument and creates a proposition.
Given a specific value of \( x \) then \( P(x) \) becomes a proposition

- The *result of applying* a propositional function \( P(x) \) to a specific object, it becomes a *proposition*. But the predicate \( P \) itself (e.g. \( P = \text{“is sleeping”} \)) is not a proposition (it is not a complete sentence).

- *E.g.* if \( P(x) = \text{“x is a prime number”} \), \( P(3) \) is the *proposition* “3 is a prime number.”

- If \( P(x) = \text{“x is sleeping”} \), then \( P(\text{Mary}) \) is the *proposition* “Mary is sleeping.”
Multiple subjects/variables

Propositional functions can be constructed with multiple subjects (i.e. multiple variables, x, y, z, etc.)

• E.g. let \( P(x,y,z) = \text{“x gave y the grade z”} \),

then if

\( x = \text{“Mike”} , y = \text{“Mary”} , z = \text{“A”} \), then

\( P(x,y,z) = \text{“Mike gave Mary the grade A.”} \)
Form a foundation for programming

• If a condition holds $P(x)$ is $T$, execute code
• Define $P(x) = "x < 10"$
• Now consider the following basic program,
  $x := 2$
  while $P(x)$ then $x = x+1$
• How many times does the while loop execute?

(Recall “:=“ means “is defined/initialized as”)
HW1 assignment uses this “:=“ operator.
Domain of Discourse
(or Universe of Discourse)

• What values can the variables take on?
  • x could be all integers, e.g. P(x) = x > 3
  • x could be all students in the class, e.g., P(x) = “student x is sleeping”

• The collection of values that a variable x can take is called x’s domain of discourse.

• For a multivariate function, P(x,y,z), the domain of discourse is the allowed values of all three x, y, and z.
• The power of a predicate function is that it lets you state things about *many* objects at once.

• E.g., let $P(x) = “x+1>x”$, where the domain of discourse is the integers.

• We can then say, “For *any* integer $x$, $P(x)$ is true” instead of $(0+1>0) \land (1+1>1) \land (2+1>2) \land ...$

• But we need quantifiers to express for what elements of the domain the predicate function creates a valid proposition.
Quantifier Expressions

- **Quantifiers** provide a notation that allows us to quantify (count) how many objects in the domain of discourse satisfy a given predicate.

- “∀” is the **FORALL** or *universal* quantifier. 
  \( \forall x \ P(x) \) means *for all* \( x \) in the u.d., \( P \) holds.

- “∃” is the **EXISTS** or *existential* quantifier. 
  \( \exists x \ P(x) \) means *there exists* an \( x \) in the u.d. (that is, 1 or more) such that \( P(x) \) is true.
The Universal Quantifier ∀

• Example:
Let the u.d. of x be parking spaces at the university.
Let $P(x)$ be the predicate "x is full."
Then the universal quantification of $P(x)$, $\forall x \ P(x)$, is the proposition:
  • “All parking spaces at UCD are full.”
  • i.e., “Every parking space at UCD is full.”
  • i.e., “For each parking space at UCD, that space is full.”
The Universal Quantifier $\forall$

$$\forall x \ P(x) \equiv P(x_1) \land P(x_2) \land P(x_3) \land \ldots \land P(x_U)$$

$P(x)$ is true for every element $x$ in $U$. 
The Existential Quantifier $\exists$

- Example:
  Let the u.d. of $x$ be parking spaces at the university.
  Let $P(x)$ be the predicate “$x$ is full.”
  Then the existential quantification of $P(x)$, $\exists x P(x)$, is the proposition:
  - “Some parking space at UCD is full.”
  - “There is a parking space at UCD that is full.”
  - “At least one parking space at UCD is full.”
The Existential Quantifier $\exists$

$$\exists x \ P(x) \equiv P(x_1) \lor P(x_2) \lor P(x_3) \lor \ldots \lor P(x_u)$$

$P(x)$ is true for at least one element $x$ in $U$. 
Free and Bound Variables

• An expression like $P(x)$ is said to have a *free variable* $x$ (meaning, $x$ is undefined).

• A quantifier (either $\forall$ or $\exists$) *operates* on an expression having one or more free variables, and *binds* one or more of those variables, to produce an expression having one or more *bound variables*. 
Example of Binding

- $P(x,y)$ has 2 free variables, $x$ and $y$.
- $\forall x \ P(x,y)$ has 1 free variable, and one bound variable. [Which is which?]

- “$P(x)$, where $x=3$” is another way to bind $x$.
- An expression with zero free variables is a bona-fide (actual) proposition.
- An expression with one or more free variables is still only a predicate: *e.g. let $Q(y) = \forall x \ P(x,y)$*
Nesting of Quantifiers

- Example: Let the u.d. of \( x \) & \( y \) be people.
- Let \( L(x, y) = "x \text{ likes } y" \) (a predicate w. 2 f.v.'s)
- Then \( \exists y \ L(x, y) = \"There is someone whom \( x \) likes.\" \) (A predicate w. 1 free variable, \( x \))
- Then \( \forall x \ (\exists y \ L(x, y)) = \"Everyone has someone whom they like.\" \) (A __________ with ___ free variables.)
Review: Predicate Logic (§1.3)

- Objects \( x, y, z, \ldots \)
- Predicates \( P, Q, R, \ldots \) are functions mapping objects \( x \) to propositions \( P(x) \).
- Multi-argument predicates \( P(x, y) \).
- Quantifiers:
  \[ \forall x \, P(x) \] : \( \equiv \) “For all \( x \)’s, \( P(x) \).”
  \[ \exists x \, P(x) \] : \( \equiv \) “There is an \( x \) such that \( P(x) \).”
- Domain of discourse, bound & free vars.
Quantifier Exercise (do this in class)

If \( R(x,y) = \text{“} x \text{ relies upon } y, \text{”} \) express the following in unambiguous English:

- \( \forall x (\exists y R(x,y)) = \) Everyone has someone to rely on.
- \( \exists y (\forall x R(x,y)) = \) There’s a poor overburdened soul whom everyone relies upon (including himself)!
- \( \exists x (\forall y R(x,y)) = \) There’s some needy person who relies upon everybody (including himself).
- \( \forall y (\exists x R(x,y)) = \) Everyone has someone who relies upon them.
- \( \forall x (\forall y R(x,y)) = \) Everyone relies upon everybody, (including themselves)!
Natural language is ambiguous!

• “Everybody likes somebody.”
  • For everybody, there is somebody they like,
    • $\forall x \exists y \text{Likes}(x,y)$
  • or, there is somebody (a popular person) whom everyone likes?
    • $\exists y \forall x \text{Likes}(x,y)$

• “Somebody likes everybody.”
  • Same problem: Depends on context, emphasis.
Still More Conventions

• Sometimes the universe of discourse is restricted within the quantification, *e.g.*, 

  \[\forall x > 0 \ P(x)\]  
  is shorthand for  
  “For all \(x\) that are greater than zero, \(P(x)\).”  
  \[= \forall x \ (x > 0 \rightarrow P(x))\]

• \(\exists x > 0 \ P(x)\) is shorthand for  
  “There is an \(x\) greater than zero such that \(P(x)\).”  
  \[= \exists x \ (x > 0 \land P(x))\]
Quantifier Equivalence Laws

• Definitions of quantifiers: If u.d.=a,b,c,...
  \( \forall x \, P(x) \iff P(a) \land P(b) \land P(c) \land ... \)
  \( \exists x \, P(x) \iff P(a) \lor P(b) \lor P(c) \lor ... \)

• From those, we can prove the laws:
  \( \forall x \, P(x) \iff \neg \exists x \, \neg P(x) \)
  \( \exists x \, P(x) \iff \neg \forall x \, \neg P(x) \)

• Which propositional equivalence laws can be used to prove this?
Negations

• We can prove the laws:
  \[ \neg \forall x \ P(x) \iff \exists x \ \neg P(x) \]
  \[ \neg \exists x \ P(x) \iff \forall x \ \neg P(x) \]
Module #1 - Logic

More Equivalence Laws

• $\forall x \forall y P(x,y) \Leftrightarrow \forall y \forall x P(x,y)$
  $\exists x \exists y P(x,y) \Leftrightarrow \exists y \exists x P(x,y)$

• $\forall x (P(x) \land Q(x)) \Leftrightarrow (\forall x P(x)) \land (\forall x Q(x))$
  $\exists x (P(x) \lor Q(x)) \Leftrightarrow (\exists x P(x)) \lor (\exists x Q(x))$

• Exercise:
  See if you can prove these yourself.
  • What propositional equivalences did you use?
Review: Predicate Logic (§1.3)

- Objects $x, y, z, \ldots$
- Predicates $P, Q, R, \ldots$ are functions mapping objects $x$ to propositions $P(x)$.
- Multi-argument predicates $P(x, y)$.
- Quantifiers: $(\forall x \ P(x)) =$ “For all $x$’s, $P(x)$.”
  $(\exists x \ P(x)) =$ “There is an $x$ such that $P(x)$.”
More Notational Conventions

- Quantifiers bind as loosely as needed: parenthesize $\forall x \ P(x) \land Q(x)$

- Consecutive quantifiers of the same type can be combined:
  $\forall x \ \forall y \ \forall z \ P(x,y,z) \iff \forall x,y,z \ P(x,y,z)$ or even $\forall xyz \ P(x,y,z)$

- All quantified expressions can be reduced to the canonical *alternating* form $\forall x_1 \exists x_2 \ \forall x_3 \exists x_4 \ldots \ P(x_1, x_2, x_3, x_4, \ldots)$
Defining New Quantifiers

- As per their name, quantifiers can be used to express that a predicate is true of any given \textit{quantity} (number) of objects.

- Define $\exists!x \ P(x)$ to mean "$P(x)$ is true of exactly one $x$ in the universe of discourse."

- $\exists!x \ P(x) \iff \exists x \ (P(x) \land \neg \exists y \ (P(y) \land y \neq x))$
  
  "There is an $x$ such that $P(x)$, where there is no $y$ such that $P(y)$ and $y$ is other than $x$."


Examples

- Can predicate logic say “there exist at least two objects with property P”?
  - Yes, that’s easy:
    \[ \exists x \exists y (P(x) \land P(y) \land x \neq y) \]
Examples ...

- Can predicate logic say “there exist exactly two objects with property P”?

  Yes:
  
  $\exists x \exists y \left( P(x) \land P(y) \land x \neq y \land \forall z \left( P(z) \rightarrow (z = x \lor z = y) \right) \right)$
Some Number Theory Examples

- Let u.d. = the natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$

- “A number $x$ is even, $E(x)$, if and only if it is equal to 2 times some other number.”
  \[ \forall x \ (E(x) \leftrightarrow (\exists y \ x=2y)) \]

- “A number is prime, $P(x)$, iff it’s greater than 1 and it isn’t the product of any two non-unity numbers.”
  \[ \forall x \ (P(x) \leftrightarrow (x>1 \land \neg \exists yz \ x=yz \land y\neq1 \land z\neq1)) \]
Calculus Example

• One way of precisely defining the calculus concept of a *limit*, using quantifiers:

\[
\left( \lim_{x \to a} f(x) = L \right) \iff \\
\left( \forall \varepsilon > 0 : \exists \delta > 0 : \forall x : \\
\left( \left| x - a \right| < \delta \right) \rightarrow \left( \left| f(x) - L \right| < \varepsilon \right) \right)
\]
Deduction Example

• Definitions:
  - s :≡ Socrates (ancient Greek philosopher);
  - H(x) :≡ “x is human”;
  - M(x) :≡ “x is mortal”.

• Premises:
  - H(s) Socrates is human. 
  - ∀x H(x)→M(x) All humans are mortal.
Deduction Example Continued

- **Some valid conclusions you can draw:**
  - $H(s) \rightarrow M(s)$  
    - Instantiate universal.  
    - *If Socrates is human then he is mortal.*
  - $\neg H(s) \lor M(s)$  
    - *Socrates is inhuman or mortal.*
  - $H(s) \land (\neg H(s) \lor M(s))$  
    - *Socrates is human, and also either inhuman or mortal.*
  - $(H(s) \land \neg H(s)) \lor (H(s) \land M(s))$  
    - Apply distributive law.
  - $F \lor (H(s) \land M(s))$  
    - Trivial contradiction.
  - $H(s) \land M(s)$  
    - Use identity law.
  - $M(s)$  
    - *Socrates is mortal.*
Bonus Topic: Logic Programming

- There are some programming languages that are based entirely on predicate logic!
- The most famous one is called Prolog.
- A Prolog program is a set of propositions ("facts") and ("rules") in predicate logic.
- The input to the program is a "query" proposition.
  - Want to know if it is true or false.
- The Prolog interpreter does some automated deduction to determine whether the query follows from the facts.
End of §1.3-1.4, Predicate Logic

• From these sections you should have learned:
  • Predicate logic notation & conventions
  • Conversions: predicate logic $\leftrightarrow$ clear English
  • Meaning of quantifiers, equivalences
  • Simple reasoning with quantifiers

• Upcoming topics:
  • Introduction to proof-writing.
  • Then: Set theory –
    • a language for talking about collections of objects.