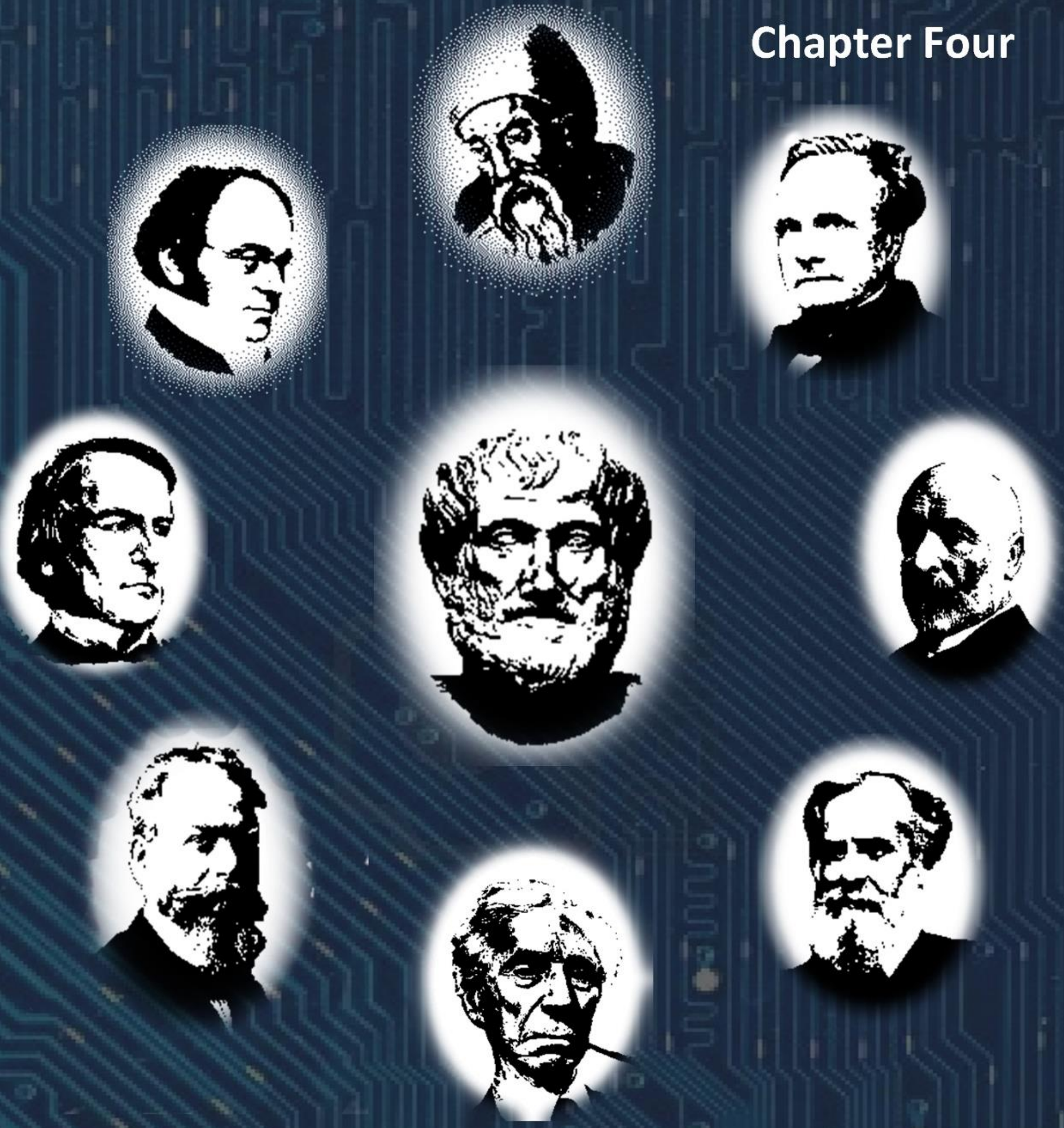
Chapter Four



REASONING

elaine rich

alan kaylor cline

Predicate Logic

The Logicians on our cover are:

Euclid (? - ?)

Augustus De Morgan (1806 – 1871)

Charles Babbage (1791 – 1871)

George Boole (1815 – 1864)

Aristotle (384 BCE – 322 BCE)

George Cantor (1845 – 1918)

Gottlob Frege (1848 – 1925)

John Venn (1834 – 1923)

Bertand Russell (1872 – 1970)

REASONING

AN INTRODUCTION TO LOGIC, SETS, AND FUNCTIONS

CHAPTER 4
PREDICATE LOGIC

Elaine Rich
Alan Kaylor Cline

The University of Texas at Austin

Image credits:

Baby pandas: http://media2.s-nbcnews.com/i/reuters/2013-07-23t094732z_1059805950_gm1e97n1db501_rtrmadp_3_china.jpg

Parrot: http://www.walterfoster.com/pages/art-tips-for-kids/18/basic-drawing-tips-techniques.html?page=4

http://www.cs.utexas.edu/learnlogic
Library of Congress Cataloging-in-Publication Data

Rich, Elaine, 1950 -
Reasoning—An Introduction to Logic Sets and Functions / Elaine Rich.— 1st ed. p. cm.
ISBN x-xxx-xxxxx-x 1

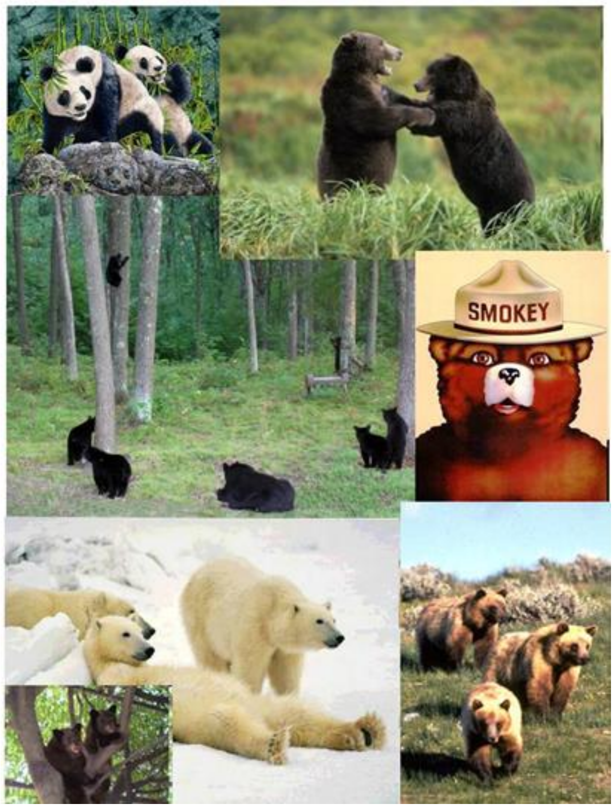
Table of Contents

Introduction, Predicates and Quantifiers

Introduction

In Boolean logic, we give names to simple statements. For example, if we wanted to talk about some bears, we could assign names as follows:

A:	Tai Shan has a tail.
S:	Smokey has a tail.
P:	Puffy has a tail.
C:	Chumpy has a tail
L:	Snowflake has a tail.



And then we might assert all of these as premises:

[1]	*A*
[2]	*S*
[3]	*P*
[4]	*C*
[5]	*L*

But now what? Can we say anything general about tail possession? For example, "If you have a tail, it can get stepped on." We'd want to be able to say that in a way that would let us conclude that each of our individual furry friends might be vulnerable to tail stomping.

Alternatively, if I tell you that Lun Lun just had twins, do you know that two more tailed creatures have entered the world? (No kidding, that's what baby giant pandas look like.)

With what we have so far, the answer to both of these questions is, "no". Our problem is that we want not just to make specific statements about specific bears (or people or whatever). We want to make general claims. For example, we want to say things like:

- *All* bears have tails.
- *Anything* that has a tail is vulnerable to it being stepped on.

Boolean logic doesn't let us do this. We need a more powerful mechanism. In this section, we'll describe such a mechanism. We'll call it ***predicate logic***.

Predicates and Quantifiers

In the system we are about to define, we'll be able to write things like this:

[1] $\forall x\ (Bear(x) \rightarrow HasTail(x))$
[2] $\forall x\ (HasTail(x) \rightarrow Stompable(x))$
[3] $\forall x\ (Bear(x) \rightarrow \exists y\ (MotherOf(y, x)))$
[4] $\forall x\ (\forall y\ (MotherOf(y, x) \vee FatherOf(y, x) \rightarrow ParentOf(y, x)))$
[5] $\forall x\ (Employee(x) \rightarrow \exists y\ (CurrentProjectOf(y, x)))$

Read these statements as follows:

[1] All bears have tails. (More literal reading of the symbols: For every object *x*, if *x* is a bear then *x* has a tail.)
[2] Anything that has a tail is stompable.
[3] Every bear has a mother. (More literal reading: For every object *x*, if *x* is a bear then there exists some object *y* such that *y* is the mother of *x*.) Notice that it's not necessarily the same mother for everyone.
[4] If you're either the mother of someone or their father, you're their parent. (More literal reading: For any objects *x* and *y*, if *y* is *x*'s mother or *x*'s father, then *y* is *x*'s parent.)
[5] Every employee is currently assigned to at least one project. (More literal reading: For every *x*, if *x* is an employee then there exists some *y* such that *y* is a current project of *x*). Notice that we've just said that there exists some project. There may exist several.

Notice a few key things about these predicate logic statements:

- There's a universe of objects that we can talk about. We'll see that that universe may be finite or (and this happens in some very important cases, like when we want to reason about numbers) the universe may be infinite.

- We assert properties (such as *Bear*, *HasTail*, *Stompable*, or *MotherOf*) about objects in the universe. Some properties (such as being a *Bear*) apply to individual objects. Others (such as *MotherOf*, *ParentOf*, and *CurrentProjectOf*) relate objects to other objects.

- We will use the same logical operators (including $\vee$, $\wedge$, $\rightarrow$, $\neg$ and $\equiv$) that we used in Boolean logic. They'll have the same meanings here.

- There are two new symbols, which we'll call ***quantifiers***:
  - $\forall$, which we'll read as, "for all". Remember this: It's an upside down A (as in **A**ll).
  - $\exists$, which we'll read as, "there exists". Remember this: It's a backwards E (as in **E**xists).

Problems

1. For each of the following claims, indicate the corresponding logical expression:

a) Everything is stompable. $\forall x\ (Stompable(x))$ $\exists x\ (Stompable(x))$

b) Some things are slimy. $\forall x\ (Slimy\ (x))$ $\exists x\ (Slimy\ (x))$

The Building Blocks of Statements

Just as in Boolean logic:

- A ***statement*** is a logical expression that has a truth value (True, abbreviated *T* or False, abbreviated *F*).

- We'll build complex logical expressions by starting with some simple building blocks and then combining them.

Two key differences between predicate logic and Boolean logic are:

- In Boolean logic, our building blocks were complete logical statements. For example, we gave the name *W* to the claim, "The sidewalks are wet." In predicate logic, we'll start with objects, like Smokey, and then build up our claims about them.

- In Boolean logic there was no way to generalize and make a single claim about many different objects (except by explicitly conjoining statements about each of them as individuals). In predicate logic, the quantifiers $\forall$ and $\exists$ will let us make such general claims in a single statement.

Now we're ready to start with the building blocks of predicate logic. We'll use:

- ***Objects*** drawn from our universe of discourse.

Examples of objects:

- *Smokey*
- *AbeLincoln*
- *Austin*
- 269

- ***Variables*** that may take on as their values any of the objects that we can talk about.

Examples of the use of variables (in these cases, the variables x and y):

- $\forall x\ (Bear(x) \rightarrow HasTail(x))$
- $\forall x\ (Bear(x) \rightarrow \exists y\ (MotherOf(y, x)))$

- ***Functions*** that apply to objects and return objects.

Examples of functions:

- *weight*(*Smokey*) will return a number that corresponds to Smokey's weight.
- *birthplace*(*AbeLincoln*) will return a location.
- *population*(*Austin*) will return a number.
- *successor*(269) will return 270.
- *plus*(4,5) will return 9. We may also write arithmetic functions in the more usual way. So (4+5) will also return 9.

- ***Predicates***, which are a special kind of function. A predicate must return a Boolean value (i.e., *T* or *F*). The job of objects and functions is to provide fodder to predicates. Predicates enable us to build logical expressions that we can reason with.

Examples of predicates:

• *Bear*(*Smokey*)	Returns True in the world of Smokey the Bear.
• *Prime*(269)	Returns True.
• *Prime*(*successor*(269))	Returns False (because 270 is not prime).
• *MotherOf*(*Gruffy*, *Smokey*)	Returns one of True or False but I don't know which. Who was Smokey's mother?

Problems

1. Consider the wff:

   $Meat(Sausage) \wedge Meat(Bacon) \wedge Diary(Milk)$

   Complete each of these statements:

   Meat is a(n): object
   variable
   function
   predicate

   Sausage is a(n): object
   variable
   function
   predicate

2. Consider the wff:

   $Prestigious(AddressOf(Kelly)) \wedge Slummy(AddressOf(Chris))$

   Complete each of these statements:

   Prestigious is a(n): object
   variable
   function
   predicate

   AddressOf is a(n): object
   variable
   function
   predicate

   Kelly is a(n): object
   variable
   function
   predicate

Defining Predicates

We define a predicate by giving a clear statement of when it is true and when it is false. As is common in English definitions, we typically write "if" when we mean "if and only if". To write predicate definitions, we use variables as placeholders.

Examples of predicate definitions:

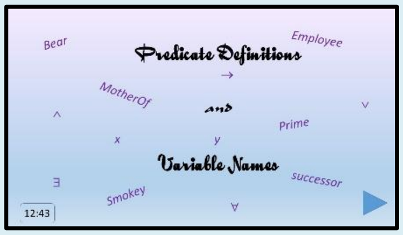
• *Bear*(x) :	True if x is a bear.
• *Prime*(x) :	True if x is a prime number.
• *MotherOf*(x, *y*) :	True if x is the mother of *y*.

Notice that we've written these definitions in English. That makes them useful for us (as people). Before we can use our predicates in a formal reasoning system, we must, of course, provide formal definitions of them. We do that with premises. We will usually need some base facts (for example, that 1 is a number or that Smokey is a bear) and some rules that enable us to derive other facts (for example, that the successor of a number is also a number or, if your mother is a bear, so are you).

Big Idea

Logic is a formal system. All it does is manipulate symbols. The names we use have no meaning to the reasoning engine. They matter only to the extent that we use them to state premises and to interpret conclusions.

https://www.youtube.com/watch?v=SXge6TRm9rM



Problems

1. Assume that Lloyd, Agnes, and Lucy are names of pets and that white, tortoise and calico are names for fur colors. We want to encode a set of facts about the fur color of our pets. We want to do that by writing:

Fur(Lucy, calico) *Fur*(Agnes, tortoise) *Fur*(Lloyd, white)

Consider the following proposed definitions for the predicate *Fur*:

I.	*Fur*(*x*, *y*) :	True if *x*'s fur color is *y.*
II.	*Fur*(*y*, *x*) :	True if *y*'s fur color is *x*.
III.	*Fur*(*x*, *y*) :	True if *y*'s fur color is *x*.

Which of these definitions is/are consistent with the way we have written our claims?

2. We want to define the predicate *gt* (for "greater than") on the integers.

(Part 1) Assume the following definition:

gt(*x*, *y*) : True if *x* is greater than *y*.

Which of the following claims is true in standard arithmetic:

a. *gt*(5, 3)
b. *gt*(3, 5)

(Part 2) Assume the following definition:

gt(*x*, *y*) : True if *y* is greater than *x*.

Which of the following claims is true in standard arithmetic:

a. *gt*(2, 1)
b. *gt*(1, 2)

Predicate Logic Well-Formed Formulas

Using these building blocks, we can now describe what a syntactically legal predicate logic expression looks like. As in Boolean logic, we'll call such an expression a ***well-formed formula*** or ***wff*** (pronounced "woof").

The simplest kind of wff is a single predicate applied to its required number of objects.

Examples of simple wffs:

[1] *Bear*(*Smokey*)	*Bear* is a predicate of one argument.
[2] *Prime*(269)	*Prime* is a predicate of one argument.
[3] *MotherOf*(*Gruffy*, *Smokey*)	*MotherOf* takes two arguments.
[4] *CurrentProjectOf*(*Shazaam*, *Chris*)	Also takes two arguments.
[5] *Bear*(x)	

The first four of our example wffs are logical statements. They have truth values. If we want to use them to reason about some situation, we'll need to choose premises so that the ones that are true can be proven to be true and the ones that are false can be proven to be false.

The last of our examples contains the variable x. We'll say that the variable x is ***unbound*** or ***free***. Before we can assign a truth value to the wff, we need to know what actual object(s) x refers to. The terminology we'll use is that we need to ***bind*** x. More about this in a minute.

Problems

1. Assume the convention that names for specific objects begin with capital letters. Variables begin with lower case letters. Which of the following logical expressions contain(s) a free (unbound) variable:

a) *Bear*(Coco)
b) *Prime*(x)
c) *Even*(24)
d) $\forall x$ (*HasSuccessor*(x))
e) $\exists y$ ($\forall x$ (*Eats*(x, y)))

We Inherit the Boolean Operators

We can build more complex wffs by applying any of the Boolean operators to one or more simpler wffs.

Examples of wffs:

[1]	$\neg Bear(Smokey)$	A statement.
[2]	$Bear(Smokey) \wedge Bear(Snowflake)$	A statement.
[3]	$Bear(x) \rightarrow HasTail(x)$	Not a statement because of unbound variable x.

As we use the Boolean operators to build larger wffs from smaller ones, we can use parentheses to group operators, just as we did in Boolean logic.

Here are two examples that use parentheses:

[4] $(Bear(Smokey) \wedge Bear(Snowflake)) \quad \vee \quad Deer(Bambi)$
[5] $Bear(Smokey) \quad \wedge \quad (Bear(Snowflake) \vee Deer(Bambi))$

Note that these two expressions are different. Because the two operators are different, parentheses matter. It's possible for one of these statements to be true and the other to be false.

Now suppose that we want to write an expression that contains many terms, all of them connected with the same operator (either *and* or *or*). For example, we might write:

[6] $((Bear(Smokey) \wedge Bear(Snowflake)) \wedge Bear(Ling\ Ling)) \wedge Bear(Tai\ Shan)$

We've been careful here (and in all our examples up until now) to use parentheses to indicate how the operators should be grouped, even though we know that Boolean *and* is associative. Grouping doesn't matter. We've proved that claim in the case of two *and*s (and three operands). So we have that:

$(p \wedge q) \wedge r$ is equivalent to $p \wedge (q \wedge r)$

We'll see later that it is straightforward (using a proof technique called induction) to prove the extension of that result to any number of operators. So, in the case where all the operators are the same, parenthesization doesn't matter. However, we'll continue to indicate a particular parenthesization until we are more confident of what we are doing.

Problems

1. For each of the following expressions, indicate whether or not it is a wff:

   a) $\mathit{Bear}(\mathit{Smokey}) \wedge \neg\neg \mathit{Deer}(\mathit{Bambi})$

   b) $\mathit{Prime}(269) \wedge \vee\ \mathit{Prime}(270)$

   c) $(\mathit{Prime}(269) \vee \mathit{Prime}(270)) \vee (\mathit{Prime}(271) \vee \mathit{Prime}(272))$

   d) $\mathit{Prime}(\mathit{age}(\mathit{Smokey}))$

2. Let's return to our example claims [4] and [5]:

   [4] $(\mathit{Bear}(\mathit{Smokey}) \wedge \mathit{Bear}(\mathit{Snowflake})) \quad \vee \quad \mathit{Deer}(\mathit{Bambi})$
   [5] $\mathit{Bear}(\mathit{Smokey}) \quad \wedge \quad (\mathit{Bear}(\mathit{Snowflake}) \vee \mathit{Deer}(\mathit{Bambi}))$

   Recall that the difference between them is the way that parentheses are used to group subexpressions.

   Consider the following situations:

   I. Smokey, Snowflake and Bambi are all deer.
   II. Smokey is a bear. Snowflake is a deer. Bambi is a bear.
   III. Smokey is a deer. Snowflake is a deer. Bambi is a bear.
   IV. Smokey is a deer. Snowflake is a bear. Bambi is a deer.

   (Part 1) Consider just claim [4]. For each of the four situations, indicate whether it makes claim [4] true or false:

   a) Situation I
   b) Situation II
   c) Situation III
   d) Situation IV

   (Part 2) Consider just claim [5]. For each of the four situations, indicate whether it makes claim [5] true or false:

   a) Situation I
   b) Situation II
   c) Situation III
   d) Situation IV

Quantifiers

Quantifiers are what give predicate logic a new kind of power that Boolean logic doesn't have.

We'll use two quantifiers:

- The ***universal quantifier*** $\forall$, which can be read as "for all". It is an upside down A (as in **A**ll).
- The ***existential quantifier*** $\exists$, which can be read as "there exists". It is a backwards E (as in **E**xists).

We can build wffs that correspond to general statements by exploiting these two quantifiers: If P is a wff, then:

$\forall x\ (P)$, and

$\exists x\ (P)$

are wffs. Any free (unbound) instance of x in P is ***bound*** by the quantifier and is then no longer free. We'll call P the ***scope*** of the quantifier. Notice that we put a pair of parentheses around P to remove any possible ambiguity about what the quantifier's scope is.

Examples of wffs with quantifiers:

- $\forall x\ (Bear(x) \rightarrow HasTail(x))$

  We have a single variable x. It is bound by the universal quantifier $\forall$. The scope of that $\forall$ is all of the wff to its right. There are now no unbound variables. So this wff is a logical statement. It's either true or false (depending on whether there are any tailless bears out there).

- $\exists x\ (USAPresident(x))$

  We have a single variable x. It is bound by the existential quantifier $\exists$. The scope of that $\exists$ is all of the wff to its right. There are now no unbound variables. So this wff is a logical statement. It happens to be true in the world in which we live.

- $\forall x\ (\forall y\ (MotherOf(y, x) \vee FatherOf(y, x) \rightarrow ParentOf(y, x)))$

  Now there are two variables, x and *y*. The variable *x* is bound by the universal quantifier $\forall$. The scope of that $\forall$ is all of the wff to its right, including the second quantifier and its scope. The variable *y* is bound by the second universal quantifier. The scope of that $\forall$ is all of the wff to its right. So we have that, for any pair of values x and *y*, if *y* is either the mother or the father of x, then *y* is also a parent of x. Both of the variables are bound, so this wff is a statement (that happens to be true).

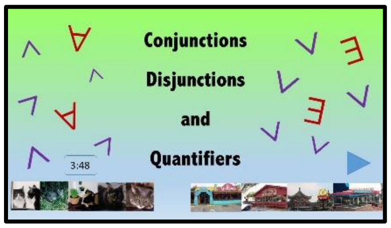
- ∀x (∃y (*MotherOf*(y, x)))

  Now there are two variables, x and *y*. The variable x is bound by the universal quantifier ∀. The scope of that ∀ is all of the wff to its right, including the second quantifier and its scope. The variable *y* is bound by the existential quantifier ∃. The scope of that ∃ is all of the wff to its right. Both of the variables are bound, so this wff is a statement (that happens to be true). Notice that the existentially quantified expression occurs *inside* the scope of the universally quantified one. This is important. It means that, given any particular x, there exists some *y* who is x's mother. Critically, we are not claiming that it's the same *y* for all possible x's. That claim, of course, would be false.

Big Idea

The scope of a quantifier is the subexpression over which it exerts control.

When the number of objects in our domain is finite, we can think of these quantifiers simply as shorthands for large Boolean expressions. In particular, let *n* be the number of objects in our domain.



https://www.youtube.com/watch?v=6GisBl2PE84

Then:

- Think of the quantifier ∀ as shorthand for a large *conjunction*. When we say:

  $\forall x\ (P(x))$, what we're really saying is:

  $$P(x_1) \wedge P(x_2) \wedge P(x_3) \wedge P(x_4) \wedge \ldots \wedge P(x_n)$$

- Think of the quantifier ∃ as shorthand for a large *disjunction*. When we say:

  $\exists x\ (P(x))$, what we're really saying is:

  $$P(x_1) \vee P(x_2) \vee P(x_3) \vee P(x_4) \vee \ldots \vee P(x_n)$$

  Remember that ∨ means *inclusive-or*, so it's possible that two or more of the disjuncts are true.

While looking at the two quantifiers, $\forall$ and $\exists$, in this way helps us to understand their meaning, it may obscure their importance. We must note:

- Even when it would be possible to write out a long conjunction or disjunction, the quantifiers can make expressions short enough that we, as people, can reason with them. There's very little chance that we could get it right if we had to write out something like this:

  $\mathit{MustPayTaxes}(\mathit{Citizen}_1) \wedge \mathit{MustPayTaxes}(\mathit{Citizen}_2) \wedge \mathit{MustPayTaxes}(\mathit{Citizen}_3) \wedge \ldots$

  But we should note that, when computers are doing the reasoning, it is possible to work with Boolean expressions with hundreds of thousands of terms. For example, this happens when reasoning about computer circuits.

- Even when it would be possible to write out a long conjunction or disjunction, the quantifiers give us a more enlightening way to say what we need to say. They capture key generalizations.

  For example, we might want to assert that, in a fair society, we all share the load. That claim isn't captured by the long conjunction shown above. It is captured by the generalization:

  $\forall x\ (\mathit{Citizen}(x) \rightarrow \mathit{MustPayTaxes}(x))$

  Similarly, when we write, $\forall x\ (\mathit{Bear}(x) \rightarrow \mathit{Mammal}(x))$, we can see an essential property of being a bear.

- Even when it would be possible to write out a long conjunction or disjunction, the quantifiers make it unnecessary to change what we're saying when our domain changes.

  Again, the tax-paying example comes to mind. We certainly don't want to have to add every new citizen by hand to the rule.

  Here's another example. Suppose that we want to write integrity constraints that describe properties that must be maintained as we work with an important database. For example, we might require that every employee have an emergency contact:

  $\forall x\ (\mathit{Employee}(x) \rightarrow \exists y\ (\mathit{EmergencyContact}(y, x)))$

  At any given time, we have a finite number of employees, so we could write this out as a long conjunction in which we state that each individual employee must have an emergency contact. But then, every time we hired someone, we'd have to go in and change that statement.

- (Most important) The quantifiers make it possible to make general claims about *infinite* domains whose elements can't be written in a finite length list. If we cared only about finite domains, we could, in principle, stick with Boolean logic. While introducing quantifiers can, in that case, make some things more convenient, it adds no formal power. Not so for infinite domains. Quantified logic lets us make claims that simply cannot be made in Boolean logic.

  You might think that you don't care about infinite domains. After all, the number of atoms in the observable universe is finite. True. But there are infinite sets that we very much want to reason about. As an example, think about numbers.

Consider the set of positive integers (an infinite set). Define:

PositiveInteger(*i*):	True whenever *i* is a positive integer.
HasSuccessor(*i*):	True whenever *i*+1 is a positive integer.

Now we can assert that every positive integer has a successor:

$$\forall x\ (\mathit{PositiveInteger}(x) \rightarrow \mathit{HasSucessor}(x))$$

This claim cannot be represented in Boolean logic.

Big Idea

Quantifiers are the reason predicate logic is more powerful than Boolean logic.

Problems

1. Define two predicates:

$P(x)$: True if x is a person.
$N(x)$: True if x is funny.

Which of the following statements is/are true in the world in which we live:

I. $\exists x\,(P(x) \wedge N(x))$
II. $\forall x\,(P(x) \wedge N(x))$
III. $\exists x\,(P(x)) \;\wedge\; \forall x\,(N(x))$

2. For each of the following predicate logic expressions, assume that the predicate names mean what they appear to mean. Think about the claim each is making. Then indicate, for each of them, one of the following:

a) A reasonable equivalent Boolean logic expression exists. What we mean here is something that could, let's say, be written in fewer than 10 lines.
b) There is, in principle, an equivalent Boolean logic expression, but it would completely crazy to try to write it out.
c) No equivalent Boolean logic expression exists.

Note: We are not asking whether the statements are true. We're only asking whether we could reasonably have represented the statements in Boolean logic.

(Part 1) $\forall x\,(Prime(x) \rightarrow \neg Prime(x+1))$
a) A reasonable Boolean logic expression exists.
b) Only an unmanageable Boolean logic expression exists.
c) No equivalent Boolean logic expression exists.

(Part 2) $\forall x\,(EmployeeOf(x, HugeMegaCorp) \rightarrow HasInsurance(x))$
a) A reasonable Boolean logic expression exists.
b) Only an unmanageable Boolean logic expression exists.
c) No equivalent Boolean logic expression exists.

(Part 3) $\forall x\,(CrayolaColorin8Box(x) \rightarrow PopularColor(x))$
a) A reasonable Boolean logic expression exists.
b) Only an unmanageable Boolean logic expression exists.
c) No equivalent Boolean logic expression exists.

(Part 4) $\forall x\,(EmailMessage(x) \rightarrow \exists y\,(RecipientAddressOf(y, x)))$
a) A reasonable Boolean logic expression exists.
b) Only an unmanageable Boolean logic expression exists.
c) No equivalent Boolean logic expression exists.

The Universe (Domain)

When we write quantified expressions, we write them with respect to some universe of discourse (sometimes called a domain).

Suppose I say:

Everyone must turn in the homework on Friday.

$\forall x\ (MustTurnInHomework(x))$

Surely I don't mean these guys:

There are two standard ways to specify the universe:

1) Define the universe at the beginning of a problem description.
2) "Guard" each claim.

To make it clear that the cats Lucy and Agnes are not required to do homework, we could:

1) Start by saying:

   Let the universe be the set of students in our class.

2) Guard our claim by saying:

   Anyone who is in our class must turn in the homework on Friday.

   $\forall x\ (InOurClass(x) \rightarrow MustTurnInHomework(x))$

Problems

1. Suppose that all we cared about was representing the fact that Lucy and Agnes don't sleep all the time. We could define two Boolean predicates:

L: Lucy (the calico) is awake sometimes.

A: Agnes (the tortoise shell) is awake sometimes.

Then we could write a simple Boolean expression:

[1] $L \wedge A$

But now suppose that we want to say something more general about cats. We'll define two predicates:

$C(x)$: True if x is a (living) cat.
$W(x)$: True if x is awake sometimes.

Which of the following statements is/are true in the world in which we live:

I. $\forall x\ (W(x))$
II. $\forall x\ (C(x) \rightarrow W(x))$
III. $\exists x\ (C(x) \rightarrow W(x))$

Quantifier Scope

Recall that the ***scope*** of a quantifier is that part of a logical expression over which the quantifier exerts control. We've already said everything we absolutely need to say about this. But, when an expression contains multiple quantifiers, it's very easy to make mistakes. So let's talk a bit more about this issue.

First, recall that we use parentheses to provide a clear indication of the scope of every quantifier. Every quantified expression has the form:

$\forall x\,(\quad)$ or $\exists x\,(\quad)$

If there are multiple quantifiers, there are multiple sets of parentheses. For example:

$\forall x\,(\forall y\,(\quad)\quad)$

Does it matter what variable names we use? Not if there's just one quantifier. So:

$\forall x\,(\ )$ is equivalent to $\forall y\,(\ )$

But if there are multiple quantifiers, we need to be careful. If one quantifier occurs within the scope of another one, we must use a different variable name for each one. If we don't, we'll probably get an expression that means something quite different from what we intended. In particular, in determining scope, we start with the innermost quantifier. Then we move outward, looking for variables that are still free and waiting to get bound.

For example, we could say that friendship is symmetric. In other words, for any two people, if *x* is friends with *y*, then *y* is also friends with x:

$\forall x\,(\forall y\,(\mathit{Friends}(x, y) \rightarrow \mathit{Friends}(y, x)))$

If we hadn't used a new variable for the second quantifier, we'd have had:

$\forall x\,(\forall x\,(\mathit{Friends}(x, x) \rightarrow \mathit{Friends}(x, x)))$

But this isn't what we're trying to say. The inner quantifier binds all occurrences of x. Friends(x, x) is true whenever someone (x) is friends with him/herself. So we have that, if you're friends with yourself, you're friends with yourself. The outer quantifier does nothing since there are no unbound instances of x for it to bind. Furthermore, we now have a relatively useless tautology instead of a possibly useful statement about what it means to be friends.

Big Idea

To be safe, always use a new variable name for each quantifier.

Problems

1. Consider: [1] $\forall x\ (\exists y\ (P(x, y)) \rightarrow Q(x))$

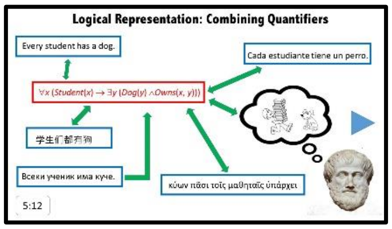
   a) What is the scope of the universal quantifier in [1]?

   b) What is the scope of the existential quantifier in [1]?

Practice Using Nested Quantifiers

To use the language of logic effectively requires being able to translate back and forth between it and ideas (possibly expressed as sentences in English or some other natural language).   It takes practice to be able to do that.

In particular, when we use nested quantifiers, the order in which we write them often (but not always – more on that later) matters.


Logical Representation: Combining Quantifiers
Every student has a dog.
Cada estudiante tiene un perro.
$\forall x\,(Student(x) \rightarrow \exists y\,(Dog(y) \wedge Owns(x, y)))$
学生们都有狗
Всеки ученик има куче.
κύων πᾶσι τοῖς μαθηταῖς ὑπάρχει
5:12

https://www.youtube.com/watch?v=XanaS0g0VlI

Problems

1. Suppose that we'd like to describe one small piece of an ideal world and say that everyone has someplace to live. Assume the domain of people. Define:

AddressOf(x, y): True if x lives at address y.

Which (one or more) of the following statements say(s) that everyone has an address:

I. $\exists x\ (\forall y\ (AddressOf(x, y)))$
II. $\forall x\ (\exists y\ (AddressOf(x, y)))$
III. $\forall x\ (\forall y\ (AddressOf(x, y)))$
IV. $\exists x\ (\exists y\ (AddressOf(x, y)))$

2. ***Bad Movie*** We want to claim that there's a movie that everyone in our class hates. Define:

Movie(x):	True if x is a movie.
InClass(x):	True if x is in our class
Hates(x, y):	True if x hates y.

Note: As you read these definitions, as well as the statements below, remember that variable names are arbitrary. We tend to reuse the same one-letter ones. The only thing that matters is that they be used consistently throughout an expression. So, for example, an equivalent definition of hates would be:

Hates(y, x): True if y hates x.

Which (one or more) of the following statements says that there's a movie that everyone in our class hates:

I. $\exists x\ (Movie(x) \wedge \forall y\ (InClass(y) \rightarrow Hates(y, x)))$
II. $\exists x\ (Movie(x) \wedge \forall x\ (InClass(x) \rightarrow Hates(x, x)))$
III. $\forall x\ (Movie(x) \rightarrow \exists y\ (InClass(y) \rightarrow Hates(y, x)))$
IV. $\forall x\ (Movie(x) \wedge \forall y\ (InClass(y)) \rightarrow Hates(y, x)))$

3. Suppose that we want to represent the fact that every pen has a color. Consider the following predicate logic expressions:

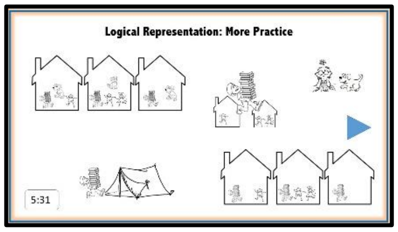
I. $\forall x\ (Pen(x) \rightarrow \exists y\ (ColorOf(y, x)))$
II. $\exists x\ (\forall y\ (Pen(x) \rightarrow ColorOf(y, x)))$
III. $\forall x\ (\forall y\ (Pen(x) \rightarrow ColorOf(y, x)))$
IV. $\exists x\ (\exists y\ (Pen(x) \rightarrow ColorOf(y, x)))$

Which (one or more) of these expressions captures our fact?

More on Using Predicate Logic

Introduction

Now we'll consider some additional issues that can arise when we want to represent claims in predicate logic.



https://www.youtube.com/watch?v=jt-B9JZAcJo

More on Scope

The details really matter when we write logical expressions that use quantifiers. So let's look at one more example of the kind of distinction that it's important that we make.

Every time we write a predicate, we need to be careful: Should we introduce a new quantifier to bind one of its variables? Or should we write the predicate inside the scope of a quantifier that we've already written?

Assume a universe that corresponds to a group of people who are planning to go on a trip. Consider the following statements:

[1] $\forall x\ (\mathit{HasArrived}(x) \rightarrow \mathit{MayBoardBus}(x))$

[2] $(\forall x\ (\mathit{HasArrived}(x))) \rightarrow \forall x\ (\mathit{MayBoardBus}(x))$

Let's read [1] carefully: It says that the rule is, for any person x, if x has arrived then x may board the bus. Good deal for the early birds.

Now let's read [2]: It says that once everyone has arrived, everyone may board the bus. The early birds will now want to strangle any stragglers.

By the way, notice that [2], while correct (assuming that it says what we're trying to say), violates our rule of thumb that tells us, that to avoid mistakes, we should use a different variable name each time we introduce a new quantifier. So, what we really should have written, just to be on the safe side, is:

[3] $(\forall x\ (\mathit{HasArrived}(x))) \rightarrow \forall y\ (\mathit{MayBoardBus}(y))$

Note, by the way, that in [2] and [3], above, we've actually used one more set of parentheses than is strictly necessary. We enclosed the complete quantified expression on the left of the implication in parentheses. We didn't really need to do that. The quantifier and its scope (already enclosed in parentheses) form a complete expression. But we must be particularly careful when a fully quantified expression is the antecedent of an implication. So we'll use the extra parentheses to help us make sure that we've written what we mean.

It may be tempting to get annoyed with our notation when it forces us to be so picky about every little parenthesis. But that's actually why the notation is great: Yes, we must be picky. But someone reading what we write knows exactly what we mean.

Big Idea

Parentheses really matter.

Problems

1. Consider the following logical expressions:

I. $\exists x\ (HasLostWallet(x) \rightarrow Frantic(x))$
II. $\exists x\ (HasLostWallet(x)) \rightarrow \exists x\ (Frantic(x))$
III. $\forall x\ (HasLostWallet(x) \rightarrow Frantic(x))$

(Part 1) Which (one or more of) these those expressions correspond(s) to the claim that if there's anyone around who's lost a wallet then there is definitely someone who's frantic:

(Part 2) Which (one or more of) of those expressions correspond(s) to the claim that anyone who has lost a wallet will be frantic?

(Part 3) Which (one or more of) those expressions correspond(s) to the claim that there exist panic-prone people who will become frantic if they lose their wallets?

When Does Quantifier Order Matter?

Does it matter in what order we write the quantifiers?

- If they are the same, often it doesn't matter.

  If we write, for example, $\forall x\ (\forall y\ ((Friends(x, y) \rightarrow Friends(y, x)))$, we're saying that for every combination of values of x and y, the statement holds. Order doesn't matter. To understand why it doesn't matter, recall that:

  - $\forall$ can be thought of as a large conjunction (expressions *and*ed together). Order doesn't matter in conjunctions since *and* is both associative and commutative.
  - $\exists$ can be thought of as a large disjunction (expressions *or*ed together). Again, order doesn't matter because *or* is both associative and commutative.

  But, to see why we can't say that it never matters, consider this claim (similar to the bus boarding claim that we just considered):

[1] $\forall x\ \underline{(\forall y\ (FinishWork(y))}\rightarrow Invited(x))$

Notice that the scope of *y* ends before the $\rightarrow$. We are saying that, for any x, if everyone finishes their work, then x will be invited to the event. In other words, if everyone finishes, everyone is invited. If we try to swap the quantifiers, we must also adjust the parentheses to make sure that every variable is bound by the appropriate quantifier. Doing that, we get:

[2] $\forall y\ \underline{(\forall x\ (FinishWork(y)}\rightarrow Invited(x)))$

Now we have that, for any *y*, if *y* finishes the work, everyone will be invited. In other words, if even one person finishes, everyone will be invited. Very different.

- If they are different, it usually does matter.

Suppose, on the other hand, that we are writing expressions over the domain of people. Then we might write:

[2] $\forall x\ (\exists y\ (Father\text{-}Of(y, x)))$

In other words, every person has a father. Notice that $\exists y\ (Father\text{-}of(y, x)$ occurs within the scope of $\forall x$. So, for any particular x, we know that there's someone who is x's father. Definitely not necessarily the same father for every x. Suppose, on the other hand, that we'd written:

[3] $\exists y\ (\forall x\ (Father\text{-}Of(y, x)))$

This says something quite different. It says that there is someone (*y*) who is the father of everyone (all possible values of x, including himself). This is patently false.

Of course, it's not that the other order is necessarily false. It's just that it produces a different statement, whose truth value may be different. There are cases where the two orders happen to produce statements with the same truth value.

Define: *Hates*(x, *y*): True if x hates *y*.

These two statements have the same truth value (in the real world):

[5] $\exists y\ (\forall x\ (Hates(x, y)))$
[6] $\forall x\ (\exists y\ (Hates(x, y)))$

Since everyone hates Flu, both [5] and [6] are true.

Big Idea

Be careful when mixing quantifiers. Order often matters. The only way to be sure that two different expressions are logically equivalent is to use sound identities and inference rules to prove that they are. In the next chapter, we'll see how to do that.

Problems

1. Suppose that we want to represent the fact that friends care about each other. Consider the following predicate logic expressions:

I. $\forall x\ (\forall y\ (Friends(x, y) \rightarrow (Friends(y, x) \wedge CaresAbout(x, y))))$
II. $\forall y\ (\forall x\ (Friends(x, y) \rightarrow (Friends(y, x) \wedge CaresAbout(x, y))))$
III. $\forall x\ (\forall y\ (Friends(x, y) \rightarrow (Friends(y, x) \wedge CaresAbout(y, x))))$

Which (one or more) of these expressions captures our fact:

2. Suppose that we want to represent the fact that somewhere within the mass of humanity, there's a pair of people who are friends. (And we'll allow for a completely narcissistic society in which it counts if you're friends with yourself.) Which of the following statements captures that fact:

a) $\forall x\ (\forall y\ (Friends(y, x)))$
b) $\forall x\ (\exists y\ (Friends(y, x)))$
c) $\exists x\ (\forall y\ (Friends(y, x)))$
d) $\exists x\ (\exists y\ (Friends(y, x)))$

3. Let's talk about cooking. Define the following predicates:

Chef(x)	True whenever x is a chef.
Dish(x)	True whenever x is a food dish.
Makes(x, y)	True whenever x makes y.

For each of the following claims, indicate the predicate logic expression that corresponds to it:

(Part 1) Every chef makes every dish.

a) $\forall x\ (\forall y\ ((Chef(x) \wedge Dish(y)) \rightarrow Makes(x, y)))$
b) $\forall x\ (Chef(x) \rightarrow \exists y\ (Dish\ (y) \wedge Makes(x, y)))$
c) $\exists y\ (Dish(y) \wedge \forall x\ (Chef(x) \rightarrow Makes(x, y)))$
d) $\exists x\ (Chef(x) \wedge (\forall y\ (Dish(y) \rightarrow Makes(x, y)))$

(Part 2) Every chef makes at least one dish. (Or how could she be a chef?)

a) $\forall x\ (\forall y\ ((Chef(x) \wedge Dish(y)) \rightarrow Makes(x, y)))$
b) $\forall x\ (Chef(x) \rightarrow \exists y\ (Dish\ y) \wedge Makes(x, y)))$
c) $\exists y\ (Dish(y) \wedge \forall x\ (Chef(x) \rightarrow Makes(x, y)))$
d) $\exists x\ (Chef(x) \wedge \forall y\ (Dish(y) \rightarrow Makes(x, y)))$

(Part 3) There's a chef who makes every dish. (Then maybe we should fire all the others.)

a) $\forall x\ (\forall y\ ((Chef(x) \wedge Dish(y)) \rightarrow Makes(x, y)))$
b) $\forall x\ (Chef(x) \rightarrow \exists y\ (Dish(y) \wedge Makes(x, y)))$
c) $\exists y\ (Dish(y) \wedge \forall x\ (Chef(x) \rightarrow Makes(x, y)))$
d) $\exists x\ (Chef(x) \wedge \forall y\ (Dish(y) \rightarrow Makes(x, y)))$

(Part 4) There are some dishes every chef makes. (Note that we actually will write this as there exists some dish that every chef makes. We won't distinguish between "at least one" and "several" or something like that.)

a) $\forall x\ (\forall y\ ((Chef(x) \wedge Dish(y)) \rightarrow Makes(x, y)))$
b) $\forall x\ (Chef(x) \rightarrow \exists y\ (Dish(y) \wedge Makes(x, y)))$
c) $\exists y\ (Dish(y) \wedge \forall x\ (Chef(x) \rightarrow Makes(x, y)))$
d) $\exists x\ (Chef(x) \wedge \forall y\ (Dish(y) \rightarrow Makes(x, y)))$

Multiple Existential Quantifiers

There's one other thing that we should say about using more than one existential quantifier in a single expression. Think of the objects that they each assert to exist as being chosen independently from each other. That means that it is possible (but not required) that they'll be the same object. So, if we write:

$$\exists x\, (\exists y\, (P(x) \wedge P(y)))$$

we are not claiming that there are two different values for which P is true. We'll see later how we can do that if we need to.

Suppose that I write:

$$\exists x\, (\exists y\, (\mathit{IsStarving}(x) \wedge \mathit{HasLotsOfFood}(y)))$$

While you might be tempted to conclude that there's one hungry person and another mean hoarder, I didn't say that. There could be just one person who's got lots of food but is saving it or is on a diet or just doesn't like the food he's got.

Problems

1. Suppose that I write:

$$\exists x\, (\exists y\, (\mathit{RedSoxFan}(x) \wedge \mathit{AstrosFan}(y)))$$

What's the smallest number of objects that must exist in my universe if this statement is true?

2. Suppose that we'd like to make the following claim about the social situation at the local school: There's both a boy and a girl whom everyone likes. Assume the domain of people.

Define:

$\mathit{Boy}(x)$:	True if x is a boy.
$\mathit{Girl}(x)$:	True if x is a girl.
$\mathit{Likes}(x, y)$:	True if x likes y.

Consider the following statements:

I. $\forall z\, (\exists y\, (\exists x\, (\mathit{Boy}(x) \wedge \mathit{Girl}(y) \wedge \mathit{Likes}(z, x) \wedge \mathit{Likes}(z, y))))$
II. $\exists x\, (\exists y\, (\forall z\, (\mathit{Boy}(x) \wedge \mathit{Girl}(y) \wedge \mathit{Likes}(z, x) \wedge \mathit{Likes}(z, y))))$
III. $\exists y\, (\exists x\, (\forall z\, (\mathit{Boy}(x) \wedge \mathit{Girl}(y) \wedge \mathit{Likes}(z, x) \wedge \mathit{Likes}(z, y))))$
IV. $\exists y\, (\forall z\, (\exists x\, (\mathit{Boy}(x) \wedge \mathit{Girl}(y) \wedge \mathit{Likes}(z, x) \wedge \mathit{Likes}(z, y))))$

How many of these statements make our claim?

One Notational Shorthand

Now that we understand when we have to be careful about the way that we write quantified expressions, we will extend our syntax slightly just to make common things less cumbersome. When we have several of the same quantifier in a row, we can collapse them like this:

$\forall x\ (\forall y\ (\forall z\ (P(x, y, z))))$ can be rewritten as $\forall x, y, z\ (P(x, y, z))$

And similarly for ∃:

$\exists x\ (\exists y\ (\exists \mathrm{z}\ (P(x, y, z))))$ can be rewritten as $\exists x, y, z\ (P(x, y, z))$

Ground Instances

A ***ground instance*** is a sentence that contains no variables.

The following statements that we've already considered are ground instances:

Bear(*Smokey*)
Prime(269)
MotherOf(*Gruffy*, *Smokey*)
CurrentProjectOf(*Shazaam*, *Chris*)

In computational logic systems, it is common to store ground instances in a different form than the one that is used for other sentences. They may be contained in a table or a database, for example.

Suppose that we want to write a set of rules that describe constraints that must be maintained as our employee database is updated. (Rules like this are called ***integrity constraints*** in the database world.) Our rules will correspond to our company policies. So we might want to write such rules as:

[1] $\forall x\ (Employee(x) \rightarrow \exists y\ (EmergencyContact(y, x)))$
[2] $\forall x\ (Employee(x) \rightarrow \exists y\ (CurrentProjectOf(y, x)))$
[3] $\forall x\ (\forall y\ (\forall z\ (\forall w\ (((Employee(x) \wedge CurrentProjectOf(y, x)$
$\wedge\ CurrentProjectOf(z, x)$
$\wedge\ CurrentProjectOf(w, x) \rightarrow ((y = z) \vee (y = w) \vee (z = w))))))$

[1] Says that everyone has someone who is their emergency contact. [2] says that, for everyone, there's at least one project that is that person's current project. [3] says that no employee may be assigned to more than two different projects. It exploits the notion of equality, which we'll say more about in the next section. For now, let's just read it. We get, "If *x* is an employee and all of *y*, *z*, and *w* are *x*'s projects, then at least two of *y*, *z*, and *w* must be the same."

To reason with these rules, we'll need some ground instances, such as:

[4] *Employee*(*Chris*)
[5] *CurrentProjectOf*(*Shazaam*, *Chris*)

But there's no need to store things like [4] and [5] as logical statements in the same framework in which we're storing [1] – [3]. They're already in the database. So we'll build a reasoning engine that can combine statements from the two systems.

Some ground instances aren't stored at all. Their truth values are computed whenever we need them. This is very common, for example when dealing with numbers.

Some examples of ground instances whose truth values would probably just be computed on demand:

Prime(937)	True	
Div(788, 3)	False	(where *Div*(x, y) means x is evenly divisible by nonzero y)

Problems

1. For each of the following, indicate whether or not it is a ground instance:

a) *EvenNumber*(7)
b) *InStateOf*(*Austin*, *Texas*)
c) $\forall x\ (\exists y\ (InStateOf(x, y)))$
d) *ToDieFor*(*chocolate*)
e) *Funny*(*x*)

What If There Aren't Any?

It's tempting to think of $\forall$ as making a stronger claim than $\exists$ does. And often it does.

For example, consider: ***Existential Bears with Tails***

[1]	$\forall x\ (Bear(x) \rightarrow HasTail(x))$	All bears have tails.
[2]	$\exists x\ (Bear(x) \wedge HasTail(x))$	There exists a bear with a tail.

[1] feels substantially stronger than [2] does. And, in fact, it says more about the real world.

But it is possible to write a universal claim that actually says *less* than the corresponding existential claim does. This happens when, in fact, there are no objects to which the universal claim applies. Sometimes, when this happens, we'll say that the universal claim is, "trivially true", in the sense that it can't be false because it doesn't apply to anything.

Let Lucy be my calico cat. Now consider two things I might say:

[3] Lucy has won every game of solitaire she's ever played.

$\forall x\ (SolitaireGame(x) \wedge Played(Lucy, x) \rightarrow Won(Lucy, x))$

[4] There is a solitaire game, played by Lucy, that she's won.

$\exists x\ (SolitaireGame(x) \wedge Played(Lucy, x) \wedge Won(Lucy, x))$

Not unsurprisingly, there are no solitaire games played and lost by Lucy. So [3] is trivially true. Yet [4], the seemingly weaker existential claim, is false since Lucy has (to my knowledge) never played solitaire.

By the way, you'll notice, in both of the examples we just showed, that while the universal claim contains an implication, the existential one does not. It has an *and* instead. Why? Suppose that we had written:

[5] $\exists x\ (Bear(x) \rightarrow HasTail(x))$

It's hard to understand what this even means in this form. Recall that an implication is true unless the antecedent is true and the consequent is false. So we can make it true simply by making the antecedent false. (We did this formally in Boolean logic with the Conditional Disjunction rule that tells us that $(p \rightarrow q) \equiv (\neg p \vee q)$. We'll soon see how to use this rule in quantified statements.) Thus we can read [5] as, "There exists something that is not a bear or that has a tail." Of course, the universe contains both many nonbears (including you) and many tailed objects (including Lucy, above). So this statement is true. But it says nothing about whether there must exist at least one actual bear that has a tail. Since that's what we were trying to talk about, we used $\wedge$ instead of $\rightarrow$.

We'll keep returning to this idea. It's very rare that what you want is an implication inside an existentially quantified statement.

Problems

1. ***The Dumb Parrot***

The owner of a pet shop tells a customer the parrot in the cage is extremely intelligent - in fact, “this bird will repeat every word he hears.” The customer, impressed, buys the parrot. In a few days, the outraged customer returns with the parrot, saying, “I spoke to him for hours every day, but this stupid bird has not repeated a single word I said.”

Which of the following is true? (Hint: the pet shop owner is also a mathematician.)

- The pet shop owner must have lied.
- There is another explanation that could take the pet shop owner off the hook for lying.

2. For each of the following statements, indicate whether or not it is true in the world in which we live:

a) All flu-causing bacteria can be killed with Diet Coke.
b) All flu cases are reported to the CDC.

Infix Predicates

So far, we've written all our predicates using what we call ***prefix notation***. In other words, we write the predicate first, followed by its arguments (in parentheses).

For example, we've written:

Bear(*Smokey*)
MotherOf(*Gruffy*, *Smokey*)
CurrentProjectOf(*Shazaam*, *Chris*)

Prefix notation is convenient when we are working with predicates with arbitrary names, and when some of our predicates (e.g, *Bear*) have just one argument, others (e.g, *MotherOf*) have two arguments, and some have three or more arguments.

But there's a common special case in which an alternative notation may be more convenient. ***Infix notation*** allows us to write binary predicates (ones with exactly two arguments) by putting a predicate symbol in between the two arguments.

The expressions in the second column are infix versions of the ones in the first column:

EQ(x, *y*)	$x = y$
NEQ(x, *y*)	$x \neq y$
GT(x, *y*)	$x > y$

We can use predicates expressed in infix notation in our logical expressions in exactly the same way we'd use prefix ones. If there's a possibility of confusion, we'll enclose them in parentheses.

Problems

1. Assume the facts of integer arithmetic. Which of the following expressions isn't true:

a) $\forall x\ (\forall y\ ((x > y) \rightarrow (x \neq y)))$
b) $\forall x\ (\forall y\ (\forall z\ (((x > y) \wedge (y > z)) \rightarrow (x \neq z))))$
c) $\forall x\ (\forall y\ (\forall z\ (((x > y) \wedge (y > z)) \rightarrow (x > z))))$
d) $\forall x\ (\forall y\ ((x \geq y) \rightarrow (x \neq y)))$
e) $\forall x\ (\forall y\ (((x = y) \wedge (y \neq z)) \rightarrow (x \neq z)))$

Equality

We just mentioned, as an example of the use of an infix predicate, the equality predicate. We will use equality in exactly the way in which you are used to using it. In particular:

Let P be any predicate. If $x = y$ and $P(x)$ is true, then we can conclude that $P(y)$ is also true.

In other words, if $x = y$, then they can be substituted for each other in any logical expression.

Suppose that we know:

[1] *Fragile*(*boxmailedbySamlastweek*)

We then discover that *boxonourdoorstep* = *boxmailedbySamlastweek*

Then we can conclude:

[2] *Fragile*(*boxonourdoorstep*)

No surprises here. So why mention it at all?

Because EQ (or =) is fundamentally different from all the other predicates we are using. The system of predicate logic that we have been describing is called ***first-order logic*** because it allows quantification over variables (and thus the objects to which they refer) but not over predicates. But we want to say that if $x = y$, then $\forall P$, P is true of both x and y or neither of them. Oops.

On the other hand, ***higher-order logics*** do allow such quantification. But we don't want to resort to them because they are logically and computationally very difficult to use.

So what to do? We can't make do without equality. So we will compromise. We will use a system called ***first-order logic with equality***. We can't, in general, quantify over predicates. But we can assert equality and use it as we would expect.

We should mention here one thing that equality is particularly useful for: It lets us say that there exists a *unique* value that satisfies some predicate. The existential quantifier, $\exists$, asserts only that there exists *at least* one such object. There might be many. If we want to claim that there is exactly one, we can assert that there exists at least one, call it x, and then assert that any such one must be equal to x.

Suppose that we want to require that every employee have exactly one emergency contact. Then we could write:

[1] $\forall x\ (\mathit{Employee}(x) \rightarrow \exists y\ (\mathit{EmergencyContact}(y, x)$
$\qquad\qquad \wedge\ \forall z\ (\mathit{EmergencyContact}(z, x) \rightarrow\ (z = y))))$

So we have that *y* is an emergency contact for x. The read the last line as, "and, for any z, if z is x's emergency contact, then z must equal *y*.

Equality also lets us say that there are at least two (or three, or whatever) *different* things that possess some property.

Suppose that we want to require that, for our manufacturing plant, every critical raw material must have at least two suppliers. Then we could write:

[1] $\forall x\ (RawMaterial(x) \wedge Critical(x) \rightarrow \exists y\ (\exists z\ (SupplierOf(y, x) \wedge$
$SupplierOf(z, x) \wedge$
$\neg(z = y))))$

Without the last line, nothing prevents both z and *y* from being bound to the same supplier. In other words, nothing prevents z = *y*.

Problems

1. Suppose that you were axiomatizing the world in which we live. Which (one or more) of the following statements that exploit equality would you give as axioms/premises:

I. $\forall x, y\ (\mathit{Co\text{-}workers}(x, y) \rightarrow (\mathit{socialSecurityNumber}(x) = \mathit{socialSecurityNumber}(y)))$
II. $\forall x, y\ (\mathit{Twins}(x, y) \rightarrow (\mathit{age}(x) = \mathit{age}(y)))$
III. $\forall x, y\ (\mathit{Married}(x, y) \rightarrow \mathit{weddingAnniversaryOf}(x) = \mathit{weddingAnniversaryOf}(y))$

2. Suppose that we want to represent our class requirement that every student work with another student on the class term project. Define:

$\mathit{Student}(x)$:	x is a student.
$\mathit{Partners}(x, y)$:	x and y are on the same team for the term project.

Consider the following claim:

$$\begin{aligned}\forall x\ (\mathit{Student}(x) \rightarrow \exists y\ (&\mathit{Student}(y) \wedge \\ &\mathit{Partners}(x, y) \wedge \\ &\neg(x = y)))\end{aligned}$$

Which of the following statements is true:

a) The claim, as stated, correctly describes the class requirement.
b) The claim, as stated, is wrong. The existential quantifier should be replaced by a universal one.
c) The claim, as stated, is too weak. There is at least one other thing that must hold for the *y* that must exist for any *x* who is a student.
d) The claim as stated, is too strong. It claims too much about what must hold for the *y* that must exists for any *x* who is a student. We should remove one of the conjuncts inside the scope of the existential quantifier.

3. Suppose that *a*, *b*, *x*, and *y* are integers and we have:

[1] $x = y$
[2] $a = b$

Assume the standard rules of arithmetic and the definition of equality that we gave above:

$$\forall P\ (\forall x, y\ (((x = y) \wedge P(x)) \rightarrow P(y)))$$

For any integer n, define: $\mathit{successor}(n) = n + 1$.

Which of the following statements is/are true:

I. $(\mathit{successor}(a) + 1) > \mathit{successor}(b)$
II. $(x > a) \rightarrow (y > b)$
III. $(a \neq x) \rightarrow ((b > x) \vee (b < x))$

Translating to and from Predicate Logic Statements

Introduction

Predicate logic, like Boolean logic, is useful because it gives us a way to represent things we care about and then to reason about them. We've already seen a lot of examples in which we have translated back and forth between English and logic.

$Prime(269)$	269 is a prime number.
$\forall x\ (Bear(x) \rightarrow HasTail(x))$	All bears have tails.
$CurrentProjectOf(Shazaam, Chris)$	Shazaam is a current project of Chris.
$\forall x\ (Pen(x) \rightarrow \exists y\ (ColorOf(y, x)))$	Every pen has a color.
$\forall P\ (\forall x, y\ ((x = y \wedge P(x)) \rightarrow P(y)))$	If x and *y* are equal, they share all properties.

In this section, we'll practice that translation skill a bit more before we move on to see how we can actually construct proofs with the statements that we've made.

Then we'll return to this issue after we've seen how to construct proofs using the statements that we write.

Getting Started – What Predicates and Objects to Use

To use the logical tools that we're describing, we first have to decide how to represent the knowledge that we have to work with. Let's call this the ***representation problem***.

In each of the examples we've done so far, we've started by defining a set of predicates. How did we pick them? How did we know that we've picked the "right" ones? Let's look at some examples.

We wanted to say that everyone has to pay their taxes. So we wrote:

$\forall x\ (Citizen(x) \rightarrow MustPayTaxes(x))$

We could have defined the *Citizen* predicate so that it specifies the country that one is a citizen of. Then we could write, for example:

$Citizen(x, USA)$

If we'd done that, then we could have, if we'd wanted to, specified different rules for different countries. We could also have broken *MustPayTaxes* apart into pieces. We could have written:

$MustPay(IncomeTax, x))$, or
$MustPay(SalesTax, x))$

In one problem, we wrote:

$\forall x\ (HasLostWallet(x) \rightarrow Frantic(x))$

But maybe we should have chosen predicates that are more general in their ability to describe losing things. Perhaps, instead of *HasLostWallet*(x), we should have written:

$HasLost(x, Wallet)$, or
$HasLost(x, y) \wedge Wallet(y) \wedge Owns(x, y)$

Note that the last one makes clear that x has lost an object that is a wallet and is, in fact, his/her own wallet.

Suppose that we want to say that everyone in the class has the flu. We could write:

$\forall x\ (InClass(x) \rightarrow HasFlu(x))$

But maybe we should choose a more general predicate that lets us talk about all kinds of diseases. Perhaps we should write:

$\forall x\ (InClass(x) \rightarrow HasDisease(x, Flu))$

In all of these examples, what's the right thing to write?

The answer is that it depends. In particular, it depends on what other things we plan to encode and how we plan to reason with all the knowledge that we have. There is no single right answer. Whenever we set about to use our logical tools to accomplish something real, we have a design problem. We need to design a vocabulary that is powerful enough to accomplish our task but not so general that we get bogged down with details that don't matter.

Big Idea

We choose predicates powerful enough for the job at hand but not more complicated than necessary.

When we're doing mathematics, we generally start with primitives that have been chosen long before us. For example, we may start with the axioms that describe the integers and arithmetic. Or the axioms of geometry. Only rarely, if we're developing a whole new theory, will we have to start from scratch choosing primitives.

When we're working with a database and describing constraints on the values it can contain, our predicates are given to us – they correspond to the fields and the values in the database. The hard representational choices have to be made when the database itself is designed.

But suppose that we want to describe more open ended problems. Suppose we want to reason about our everyday world. Then what generally happens is that we'll start with one set of predicates and objects. We'll begin writing down what we know and seeing how we can reason with what we've written. We'll discover that what we've got isn't general enough for some new thing we want to say. So we'll change some things and try again.

Problems

1. Suppose that, instead of writing *HasLostWallet*(*x*), we'd chosen to describe the lost wallet situation by writing:

$$\forall x, y\ ((HasLost(x, y) \wedge Wallet(y) \wedge Owns(x, y)) \rightarrow Frantic(x))$$

Then we could also represent some other facts about people who lose things.

(Part 1) Suppose we want to prove that people get frantic if they lose any important thing. This wouldn't have been straightforward using our original representation. But now it should be.

Consider the following facts that might be relevant:

I. $\forall x\ (Wallet(x) \rightarrow Important(x))$
II. $\forall x\ (\ (\exists y\ (HasLost(x, y) \wedge Important(y))) \rightarrow Frantic(x))$
III. $\forall x\ (\exists y\ ((HasLost(x, y) \vee Important(y)) \rightarrow Frantic(x)))$

Which combination of these facts would help us to prove our conclusion:

(Part 2) So now we've seen how a more general representation could be useful. Next let's see whether we've gone far enough. We've got the predicate *Important*(*x*). Consider the following sentences that we might like to translate into logic:

I. People get frantic when they lose something that is important to them (but not generally when it's important only to someone else).
II. People get frantic when they lose something that is both important and expensive to replace.
III. People get frantic when they lose something that is super, super important.

Which of these statements can we encode with the representation we've got for describing importance? (Hint: Try to encode each of these statements. You may need some additional predicates for new concepts. But try to stick with what we've got so far for representing importance.)

Functions or Predicates?

Let's now look at one particular kind of representational decision that we often have to make. By now, you may have noticed that we have two different mechanisms for indicating relationships between objects:

- Predicates, such as *MotherOf*(*Gruffy*, *Smokey*). We've used this to say that Gruffy is the mother of Smokey.

- Functions, such as *weight*(Smokey) and *age*(Smokey). Using these functions, we can say things like *weight*(Smokey) = 567 or *age*(Smokey) = 70.

You may be wondering what the difference is between a function and a predicate. In particular, couldn't we equally well have:

- used a *motherof* function. Then we could write, *motherof*(Smokey) = Gruffy.

- used *Weight* and *Age* predicates. Then we could have that *Weight*(*Smokey*, 567) and *Age*(Smokey, 70) would be true.

Yes, in all these cases we could have used either technique.

In general, the one big difference is that a function must return exactly one value. So we can use a *motherof* function:

- *motherof*(*Smokey*) = *Gruffy*, because Smokey has exactly one mother.

But we cannot write:

- *childof*(*Gruffy*) = ___, because, while we have that Smokey is Gruffy's child, we assume there are probably more. Which one should be the value of this expression?

So, to represent claims about Gruffy's children, we must use a predicate. Then we can write:

- *ChildOf*(*Smokey*, *Gruffy*)
- *ChildOf*(*Piper*, *Gruffy*) (in case Gruffy had a second child, Piper)

In computational logic systems, the decision of when to use functions and when to use predicates is usually made on the basis of various practical considerations. When we're writing out our claims, we'll use whichever makes it easier to see what is going on.

Problems

1. Suppose that we are a wholesaler (we sell to other companies) and we want to talk about the customers we have for our various products. Which one or more of the following describe(s) our options:

I. We could use a predicate *Buys*(*company*, *product*). We'll read such a predicate as saying that *company* buys *product* from us. We would assert specific claims like, *Buys*(*YesCorp*, *Ladders*) or *Buys*(*PaintersAreUs*, *Ladders*) or *Buys*(*PaintersAreUs*, *Tarps*).

II. We could use a function *whoareourcustomers*(*product*). So we could write, for example, *whoareourcustomers*(*Ladders*) and expect to get back the name of the customer for ladders.

III. We could use a function *whatdotheybuy*(*customer*). So we could write, for example, *whatdotheybuy*(*PaintersAreUs*) and expect to get back the product that *PaintersAreUs* buys.

Definitions

Mathematics seeks to provide a basis for rigorous and insightful arguments. No surprise then that mathematicians make use of the logical tools we've described. They start with definitions, then assert axioms, and move on to claims that they can show to be theorems. We too will start with definitions.

A ***definition*** (like this one) is a claim that one thing (the thing being defined) is equivalent to some other thing (typically described in terms that we've already defined). So definitions generally take the form of logical equivalence statements.

Divisibility

Let's do a simple definition first. Our domain is the set of integers (including zero and the negative integers). We will say that *Div*(x, *y*) is true if x is evenly divisible by nonzero *y*:

[1] $\forall x, y\ (Div(x, y) \equiv ((y \neq 0) \wedge \exists z\ (x = y \cdot z)))$

We can read this as, "Saying that x is divisible by nonzero *y* is equivalent to saying that x is the product of *y* and some integer z."

Even numbers

Next, we can use our definition of *Div* to define what it means for an integer to be even:

[2] $\forall x\ (Even(x) \equiv Div(x, 2))$

We can read this as, "Saying that an integer x is even is equivalent to saying that it is divisible by 2."

In these last two examples, you saw that we read our definitions as:

> Saying that <one thing > is equivalent to saying <something else>.

It's more common to read ≡ as "if and only if". So we can read our second definition as:

> *x* is even if and only if …

Since "if and only if" gets a bit cumbersome, it can be abbreviated ***iff***.

Finally, we should probably warn you here: Mathematicians tend, in definitions, to say:

> *x* is <something significant> *if* <certain significant things are true>.

Note that we've written just "if" when we actually meant "if and only if". You may have noticed that we've been using this convention also (including in our definition of *Div* in this very example). It's very common. Just keep in mind that the English word "if", when used in a definition, generally means "if and only if".

Big Idea

Mathematical definitions are written as logical statements that exploit an equivalence statement.

Prime Numbers

Now let's define what it means to be a prime number. We'll consider the domain of positive integers. Recall that a prime number is a positive integer greater than 1 that is not evenly divisible by any positive integer except itself and 1. So, for example:

- Some prime numbers are: 2, 3, 5, 7, 11, 257, 463, and 997.
- Some nonprime numbers are: 4, 6, 8, 9, 10, 12, 15, 27, 201, 403, 819, and 931.

We can now state clearly what it means to be a prime. We'll use the *Div* predicate that we defined above and we'll exploit the notion of equality that we've already described.

Assume that the universe is the positive integers. Here's one definition:

[3] $\forall x\ (\mathit{Prime}(x) \equiv (\ (x > 1) \wedge \forall y\ (\mathit{Div}(x, y) \rightarrow ((x = y) \vee (y = 1)))))$

We can read what we've written as: x is Prime if and only if:
- x is greater than 1, and
- For any *y* that evenly divides x, *y* is 1 or it is x itself.

So, for example, 6 isn't prime because it's divisible by 2, which is not equal to 6 and not equal to 1.

Here's another, equivalent definition:

[4] $\forall x\ (\mathit{Prime}(x) \equiv (\ (x > 1) \wedge \neg\exists y\ (\mathit{Div}(x, y) \wedge \neg(x = y) \wedge \neg(y = 1))))$

Now we can read what we've written as: x is prime if and only if:
- x is greater than 1, and
- There exists no *y* that has all three of these properties:
    - It evenly divides x.
    - It isn't equal to x.
    - It isn't 1.

Nifty Aside

If you want to experiment with finding prime numbers, visit http://mathforsmartypants.com/resources/primecheck.php for a nice primality checking app.

Of course, once we've defined predicates, like *Prime* and *Even*, we can use them to make formal claims that we can then set about to prove.

Let's write a logical expression that corresponds to the claim that the only even prime number is 2 (a true claim that is quite easy to see: No prime number is less than 2. And any larger even number has 2 has a divisor and so is not prime.)

We state our claim as:

[5] $\forall x\ ((\mathit{Even}(x) \wedge \mathit{Prime}(x)) \rightarrow (x = 2))$

Problems

1. We wish to assert the claim that there is no prime number strictly between 317 and 337. (Note that "strictly between" means that the required number may not be either 317 or 337.)

(Part 1) Which one or more of the following correctly asserts this claim:

I. $\forall x\ (\mathrm{Prime}(x) \rightarrow \neg(\ (x < 317) \wedge (x > 337)))$
II. $\forall x\ (\mathrm{Prime}(x) \rightarrow (\ (x \leq 317) \wedge (x \geq 337)))$
III. $\forall x\ (\mathrm{Prime}(x) \rightarrow (\ (x \leq 317) \vee (x \geq 337)))$

(Part 2) Is this claim true?

2. ***Composite Numbers*** A composite number is an integer, greater than 1, that is not prime. Without using the predicate *Prime* that we've already defined (because the point here is to get practice at the definition process), we want to give a formal definition of what it means to be composite. Call our new predicate *Composite*. Assume the domain of positive integers. Which one or more of the following correctly defines *Composite*:

I. $\forall x\ (Composite(x) \equiv (\neg(x = 1) \wedge \forall y\ (Div(x, y) \wedge \neg(x = y) \wedge \neg(y = 1)\ )\ )\ )$
II. $\forall x\ (Composite(x) \equiv (\neg(x = 1) \wedge \exists y\ (Div(x, y) \wedge \neg(x = y) \wedge \neg(y = 1)\ )\ )\ )$
III. $\forall x\ (Composite(x) \equiv (\neg(x = 1) \wedge \exists y\ (Div(x, y) \vee \neg(x = y) \vee \neg(y = 1)\ )\ )\ )$

Negation

While negation is a straightforward concept, it's easy to make mistakes when encoding it in logic. So we'll go through some examples for practice.

Recall the ***Both My Pills*** example that we considered back in the introduction to this course: Consider the claim, "I didn't take both of my pills this morning." We pointed out that this sentence could mean either of these things:

[1] For each of my pills, it's the case that I didn't take it this morning.
[2] I took one of my pills this morning, but not both of them.

Now we can show how to encode each of these meanings in predicate logic. Define:

Pill(x): True if x is one of my pills.
Took(x, *w*): True if I took x at time *w*.

For both meanings, we need to say (if it's important to do so) that I take exactly two pills. We can do that with the conjunction of two claims:

[3] $\exists x, y\ (Pill(x) \wedge Pill(y) \wedge \neg(x = y))$ I have at least two pills.
[4] $\forall x, y, z\ ((Pill(x) \wedge Pill(y) \wedge Pill(z)) \rightarrow ((x = y) \vee (x = z)))$ I have no more than two pills.

[3] can be read literally as, "There exist two objects that are my pills and that are not the same thing."
[4] can be read literally as, "If there are three objects that are my pills, there is some pair of them that are the same thing.

This may feel like a very clunky way of saying that I have exactly two pills. It is. Unfortunately, to do it more elegantly, we need to develop a language for talking about sets (in this case, the set of my pills) and the sizes of sets. We'll do that in the follow on course on Sets, Functions and Relations.

Now on to encoding the specific claims [1] and [2] (either of which can be *anded* with the conjunction of [3] and [4]):

[1] $\forall x\ (Pill(x) \rightarrow \neg Took(x, ThisMorning))$
[2] $\exists x\ (Pill(x) \wedge Took(x, ThisMorning)) \wedge \exists y\ (Pill(y) \wedge \neg Took(y, ThisMorning))$

We make negated statements much more frequently than you might imagine.

English Aside

Recall that, in English, there are many ways to make negative statements. We can use words that contain implied negations, such as, "never", "nowhere", "none", "only", "just", "unless", "except", "without", and "but". We can also use negative prefixed, such as, "un-", "il-", and "in-".

Lying Suppose that we want to encode the fact that some people never lie. Define:

Person(x):	True if x is a person.
Time(*t*):	True if *t* is a time.
Lying(x, *t*):	True if x is lying at time *t*.

Notice that the word "never" contains an implied negation. So we can write:

[1] $\exists x\ (Person(x) \wedge \forall t\ (\neg Lying(x, t)))$

We read [1] as: There's some x who is a person and, at all times, x isn't lying at that time.

Sometimes we can say the same thing in two different ways, one that involves negation and one that doesn't. It may depend on the predicates we choose to work with.

Let's return to the ***Lying*** problem. This time, define:

Person(x):	True if x is a person.
Time(*t*):	True if *t* is a time.
TruthTelling(x, *t*):	True if x is telling the truth at time *t*.

Now we can write:

[2] $\exists x\ (Person(x) \wedge \forall t\ (TruthTelling(x, t)))$

Suppose that we add a new premise:

[3] $\forall x,y\ (TruthTelling(x, t) \equiv \neg Lying(x, t))$

Then we should (using the proof techniques we're about to discuss) be able to prove [1] from [2] and vice versa.

When we discuss the new inference rules that we'll need for working with predicate logic statements, we'll formalize what happens with existential and universal quantifiers when they are negated. For now, let's just look at a simple example where we can easily see how to encode what we want to say.

Suppose that we want to encode the claim that no one likes beets. (This is apparently not true, although some of us find that amazing.) In any case, we can encode things, whether they're true or not. Define:

Likes(x, y): True if x likes y.

There are two probably equally obvious ways to say what we have to say:

[1] $\forall x\ (\neg Likes(x, Beets))$ Everyone dislikes beets. or

[2] $\neg\exists x\ (Likes(x, Beets))$ There doesn't exist anyone who likes beets.

Problems

1. Define:

$Curable(x)$:	True if x is a curable disease.

Which one or more of the following logical expressions corresponds to the claim that leprosy is not an incurable disease?

I. $Curable(Leprosy)$

II. $\neg Curable(Leprosy)$

III. $\neg\neg Curable(Leprosy)$

2. Define:

$Lying(x, t)$:	True if x is lying at time t.
$Wednesday(t)$:	True if t is a Wednesday

Which one or more of the following logical expressions corresponds to the claim that John only lies on Wednesday? (Notice that the word "only" contains an implied negation.)

I. $\forall t\, (\neg Wednesday(t) \rightarrow \neg Lying(John, t))$

II. $\neg\exists t\, (\neg Wednesday(t) \wedge Lying(John, t))$

III. $\forall t\, (Wednesday(t) \rightarrow Lying(John, t))$

3. Let's do one more example that involves our corporate database and our corporate policies. We want to encode the fact that only supervisors can sign time cards. Define:

$Super(x)$:	True if x is a supervisor.
$CanSign(x, y)$:	True if x can sign y.
$Timecard(x)$:	True if x is a timecard.

Which one or more of the following expressions corresponds to our claim:

I. $\forall x, y\, ((Super(x) \wedge Timecard(y)) \rightarrow CanSign(x, y))$

II. $\forall x, y\, ((\neg Super(x) \wedge Timecard(y)) \rightarrow \neg CanSign(x, y))$

III. $\forall x, y\, ((Super(x) \wedge \neg Timecard(y)) \rightarrow CanSign(x, y))$

4. Consider one famous girl of song, my girl, "the girl who loves nobody but me." Define:

$Loves(x, y)$:	True if x loves y.

Which one or more of the following expressions describes this girl, given the reading most of us have of the song:

I. $Loves(MyGirl, Me) \wedge \forall y\, (\neg(y = Me) \rightarrow \neg Loves(MyGirl, y))$

II. $Loves(MyGirl, Me) \wedge \forall y\, ((y = Me) \vee \neg Loves(MyGirl, y))$

III. $\forall y\, ((y = Me) \vee \neg Loves(MyGirl, y))$

Validity, Satisfiability, Contradiction and Counterexamples

Introduction

Recall that a statement is a wff that has a truth value. All of the following are statements:

[1] $Bear(Smokey)$
[2] $\forall x\ (Bear(x) \rightarrow Animal(x))$
[3] $\forall x\ (Animal(x) \rightarrow Bear(x))$
[4] $\forall x\ (Animal(x) \rightarrow \exists y\ (MotherOf(y, x)))$
[5] $\forall x\ ((Animal(x) \wedge \neg Dead(x)) \rightarrow Alive(x))$

The following wffs are not statements because they contain unbound variables:

[6] $Bear(x)$
[7] $Animal(x) \vee Vegetable(x)$

We don't have any way to assign truth values to these wffs without changing them in some way. (We could substitute a particular value, such as Smokey, for *x*. Or we could enclose the statement inside a quantifier.) But then we'd be assigning meaning to a different wff.

So we'll focus here just on statements. Returning to [1] – [5] above, we observe that [1], [2], [4], and [5] are true in our everyday world (assuming the obvious referent for the constant *Smokey* and the obvious meanings of the predicates *Bear*, *Animal*, and *MotherOf*). On the other hand, [3] is not true.

Meaning

Recall the examples from the previous slide:

[1] $Bear(Smokey)$
[2] $\forall x\ (Bear(x) \rightarrow Animal(x))$
[3] $\forall x\ (Animal(x) \rightarrow Bear(x))$
[4] $\forall x\ (Animal(x) \rightarrow \exists y\ (MotherOf(y, x)))$
[5] $\forall x\ ((Animal(x) \wedge \neg Dead(x)) \rightarrow Alive(x))$

As these examples show, determining whether a sentence is true requires appeal to the *meanings* of the constants, functions, and predicates that it contains. (This must be so because [2] and [3] have exactly the same structure. It's just their words that are different. Yet one is true and one is false.) An ***interpretation*** for a sentence *w* assigns meanings to the symbols of *w*. How do we specify an interpretation? How do we know that [2] is true and [3] isn't? So far, we've been appealing to what we, as people, think the words that we've chosen for our objects and predicates "mean" in English. But we need more than that.

Our goal, when we use any logical system, is to get at truth. So what we do is to choose premises that assert things that are true. We write those premises in terms that we pick. We can pick any terms we like as long as we use them consistently to describe the situation that we want to reason with. The logical system doesn't care what names we use. It applies its inference rules and generates new statements that it asserts to be true. But then, and here's the key: When we read off these new statements, we do so using the same terminology that we started with. As long as we do that, we'll get true statements that make sense to us.

Let's return to the question of what an interpretation has to do. In Boolean logic, all it has to do is to assign a meaning to each variable. There are only two possible meanings, *T* and *F*. An interpretation of a formula then is just a single row of the formula's truth table. That row assigns a meaning to each of the variables and then to each of the subexpressions, ending with the whole formula. Every time we wrote a truth table, what we were really doing was writing out all possible interpretations of the formula we were working with. Then we could say that:

- A Boolean formula is valid (or is a tautology) if it is true in all interpretations. In other words all rows of its truth table end in *T*.

- A Boolean formula is satisfiable if there exists some interpretation in which it is true. In other words, at least one row of its truth table ends in *T*.

- A Boolean formula is unsatisfiable (alternatively, it is a contradiction) if there exists no interpretation that makes it true. In other words, every row of its truth table ends in *F*.

Validity and Satisfiability in Predicate Logic

Are there analogous definitions for predicate logic sentences? When we start to write proofs (which we're about to do), we'll want to be able to say that we've proved that a sentence must be true "no matter what". In other words, that it's valid.

We can start the same way:

- A predicate logic sentence *w* is ***valid*** if it is true in all interpretations. In other words, it is true regardless of what the constant, function, and predicate symbols "mean".
- A predicate logic sentence *w* is ***satisfiable*** if there exists *some* interpretation in which *w* is true.
- A predicate logic sentence is ***unsatisfiable*** (i.e., it is a ***contradiction***) if it is not satisfiable (in other words, there exists no interpretation in which it is true). As before, we note that any sentence *w* is valid just in case $\neg w$ is unsatisfiable.

But now things change. There is no longer any guarantee that we can write out a finite sized truth table that enumerates all the possible interpretations. It's true that we'll only have a finite set of predicates. And each predicate, when applied to a particular value or set of values (for example, *Bear*(*Smokey*)) must evaluate to one of the two values, *T* or *F*. But there may be a very large, or even infinite, domain from which the values can be chosen. And what the predicate evaluates to typically depends on the values to which it's applied. For example, we'll generally assert that *Bear*(*Smokey*) is true but *Bear*(*Bambi*) is not.

So, while validity and satisfiability are as important now as they were in Boolean logic, the techniques that we'll use to prove them will have to be different. What we'll see, in the next section of this course, is that while we can't build on the truth table techniques that served us well in Boolean logic, we can build on the natural deduction approach that we described at the end of our Boolean logic discussion.

Consider:

[1] $\forall x\ (P(x) \rightarrow (P(x) \vee Q(x)))$

It's straightforward, in Boolean logic, to show that $p \rightarrow (p \vee q)$ is a tautology. We'll import into our predicate logic reasoning engine all of the rules that let us do that. Then we'll add new rules that will let us argue that this claim must be true for any individual x and so it must be true for all x. This is so, regardless of what *P* or Q means. So [1] is a tautology. Alternatively, it is valid. It is, of course, also satisfiable since, being true in all interpretations, it is true in at least one of them.

Next, consider:

[2] $\neg(\forall x\ (P(x) \vee \neg P(x)))$

Again, let's exploit our Boolean inference rules. For any individual x, either *P* is true of it or ¬*P* is true of it (remember the Law of the Excluded Middle). So, without the negation, [2] would be valid. Since that means it's true in all interpretations, the negation of it cannot be true in any. So [2] is unsatisfiable.

Finally, consider:

[3] $\forall x\ (P(x, x))$

[3] is not valid but it is satisfiable. Suppose that the universe is the integers and *P* is the predicate *LessThanOrEqualTo*. Then *P* is true for all values of x. But, again with the integers as the universe, suppose that *P* is the predicate *LessThan*. Now *P* is false for all values of x (you can't be less than yourself). Finally, let the universe be the set of all people and let *P* be the predicate *HasConfidenceInTheAbilityOf*. Now *P* is true of some values of x (i.e., of those people who have self-confidence) and false of others.

Problems

1. Consider: [1] $\forall x\ (P(x) \vee Q(x))$

   Is [1]:

   a) Valid
   b) Satisfiable but not valid
   c) Unsatisfiable

2. Consider: [2] $\exists x\ (P(x) \wedge \neg P(x))$

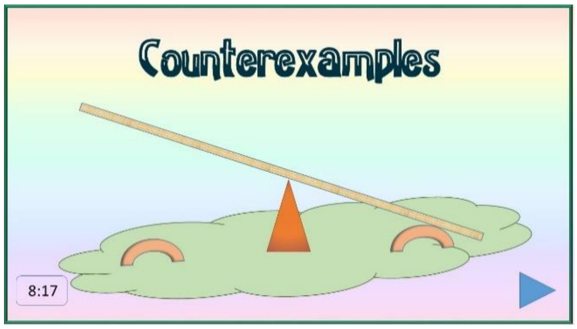
   Is [2]:

   a) Valid
   b) Satisfiable but not valid
   c) Unsatisfiable

Counterexamples

We are about to present a collection of techniques for showing that a claim must be true in *all* interpretations. (Or, similarly, that it is false in *all* interpretations.) If there are infinitely many such interpretations, we need something other than "try them all".

But suppose that we want to show that some claim is *not* valid. Or that it is *not* a contradiction. Then it suffices to find a single interpretation that makes a liar out of it. We'll call such an interpretation a ***counterexample***. Universally quantified statements are typically the easiest to refute in this way.



https://www.youtube.com/watch?v=oJGraKc1Sjc

Also note that we can refute the claim that an expression is a contradiction by showing a single set of values that make the expression true.

Let the universe be the rational numbers. Now consider:

$$\forall x, y\ (\exists z\ (z = x/y))$$

We can show that this claim is not a tautology by finding a single (x, y) pair of values such that the required z does not exist.

That is easy. Let $y = 0$. (It doesn't matter what x is.)

Problems

1. Consider the following set of claims:

[1] $\forall x\ (Cat(x) \rightarrow Loves(x, Catnip))$
[2] $\forall x\ (Loves(x, Catnip) \rightarrow Cat(x))$
[3] $\forall x\ (Loves(x, Chocolate) \rightarrow Loves(x, IceCream))$
[4] $\forall x\ ((Cat(x) \wedge Calico(x)) \rightarrow Loves(x, Chocolate))$
[5] $\forall x\ (Cat(x) \rightarrow \neg Dog(x))$

Let *ModelOfCatsAndDogs* be the conjunction of these five claims.

For each of the following statements, indicate whether or not it is a counterexample to *ModelOfCatsAndDogs*:

(Part 1) $Cat(Lucy) \wedge Calico(Lucy) \wedge \neg Loves(Lucy, Icecream)$
(Part 2) $Cat(Agnes) \wedge Loves(Agnes, Paper)$
(Part 3) $Dog(Scarlet) \wedge \forall x\ (Loves(Scarlet, x))$
(Part 4) $\mathrm{Dog}(Lloyd) \wedge Loves(Lloyd, Chocolate))$

2. ***Zamzows and Tenockritus*** For the purpose of this question, assume that the zamzow is a newly discovered species and tenockritus is an ailment that afflicts species like the zamzow. For each of the following claims, indicate an assertion that, if true, would *falsify* the claim:

(Part 1) $\forall x\ (Zamzow(x) \rightarrow HasTenockritus(x))$ All zamzows have tenockritus.

a) Alfred and Reggie are zamzows that have tenockritus.
b) There are no zamzows.
c) Clora isn't a zamzow but has tenockritus.
d) Peapod is a zamzow and, by the way, no one has tenockritus.
e) No one has tenockritus.

(Part 2) $\exists x\ (Zamzow(x) \wedge HasTenockritus(x))$ Some zamzows have tenockritus.

a) Alfred and Reggie are zamzows that have tenockritus.
b) Peapod is a zamzow who does not have tenockritus.
c) Chlora isn't a zamzow but has tenockritus.
d) Tiddlywinks is the only zamzow and does not have tenockritus.
e) Only bears have tenockritus.

(Part 3) $\neg(\exists x\ (Zamzow(x) \wedge HasTenockritus(x)))$ No zamzow has tenockritus.

a) Alfred and Reggie are zamzows that have tenockritus.
b) Peapod is a zamzow who does not have tenockritus.
c) Chlora isn't a zamzow but has tenockritus.
d) Tiddlywinks is the only zamzow and does not have tenockritus.
e) Only bears have tenockritus.