Now we're going to start talking about first-order logic, which extends propositional logic so that we can talk about things.

In propositional logic, all we had were variables that stood, not for things in the world or even quantities, but just facts, Boolean statements that might or might not be true about the world, like whether it's raining, or greater than 67 degrees; but you couldn't have variables that stood for tables or books, or the temperature, or anything like that. And as it turns out, that's an enormously limiting kind of representation.

In first-order logic, variables refer to things in the world and you can quantify over them. That is, you can talk about all or some of them without having to name them explicitly.
There are lots of examples that show how propositional logic is inadequate to characterize even moderately complex domains. Here are some more examples of the kinds of things that you can say in first-order logic, but not in propositional logic.

"When you paint the block it becomes green." You might have a proposition for every single aspect of the situation, like "my book is black" and "my book is green" and "my book is red", and then you could say that if my book was black and I paint it then after that my book is green. But you'd have to have one of those propositions for every single book, or every single desk, or every single thing in the world. You couldn't say that, as a general fact, after you paint something it becomes green.

Let's say you want to talk about what happens when you sterilize a jar. It kills all the bacteria in the jar. Now, you don't want to have to name all the bacteria; to have to say, bacterium 57 is dead, and bacterium 93 is dead. Each one of those guys is dead. All the bacteria are dead now. So you'd like to have a way not only to talk about things in the world, but to talk about all of them, or some of them, without naming any of them explicitly.

In the context of providing flexible computer security, you might want to prove or try to understand whether someone should be allowed access to a web site. And you could say: a person should have access to this web site if they've been personally, formally authorized to use this web site or if they are known to someone who has access to the web site. So you could write a general rule that says that and then some other system or this system could try to prove that you should have access to the web site. In this case, what that would mean would be going to look for a chain of people that are authorized or known to one another that bottoms out in somebody who's known to this web site.
First-order logic lets us talk about things in the world. It's a logic like propositional logic, but somewhat richer and more complex. We'll go through the material in the same way that we did propositional logic: we'll start with syntax and semantics, and then do some practice with writing down statements in first-order logic.

The big difference between propositional logic and first-order logic is that we can talk about things, and so there's a new kind of syntactic element called a term. And the term, as we'll see when we do the semantics, is a name for a thing. It's an expression that somehow names a thing in the world. There are three kinds of terms:

- Constant symbols. They are names like Fred or Japan or Bacterium39. Those are symbols that, in the context of an interpretation, name a particular thing.

- Variables. They are not really syntactically differentiated from constant symbols. We'll use capital letters to start constant symbols (think of them as proper names), and lower-case letters for term variables. (It's important to note, though, that this convention is not standard, and in some logic contexts, such as the programming language Prolog, they adopt the exact opposite convention).
Slide 5.1.12
The last kind of term is a function symbol, applied to one or more terms. We'll use lower-case for function symbols as well. So another way to make a name for something is to say something like "f(x)". If "f" is a function, you can give it a term and then f(x) names something. So, you might have mother-of(John) or f(f(x)). These three kinds of terms are our ways to name things in the world.

Slide 5.1.13
In propositional logic we had sentences. Now, in first-order logic it's a little bit more complicated, but not a lot. So what's a sentence? There's another kind of symbol called a predicate symbol. A predicate symbol is applied to zero or more terms. Predicate symbols stand for relations, so we might have things like On(A,B) or Sister(Jane, Joan). "On" and "Sister" are predicate symbols; "a", "b", "Jane", "Joan", and "mother-of(John)" are terms.

Slide 5.1.14
A sentence can also be of the form "t1 = t". We're going to have one special predicate called equality. You can say this thing equals that thing, written term, equal-sign, term.

Slide 5.1.15
There are two more new constructs. If v is a variable and Phi is a sentence then (upside-down-A v . phi), and (backwards-E v . phi) are sentences. You've probably seen these symbols before informally as "for all" and "there exists", and that's what they're going to mean for us, too.
Slide 5.1.16
Finally we have closure under the sentential operators that we had before, so you can make complex sentences out of other sentences using and, or, not, implies, equivalence, and parentheses, just as before in propositional logic. All that basic connective structure is still the same, but the things that we can say on either side have gotten a little bit more complicated.

All right, that's our syntax. That's what we get to write down on our page.

6.034 Notes: Section 5.2

Slide 5.2.1
We're going to do the semantics informally. This isn't really going to look informal to you, but compared to the sorts of things that logicians write down, it's pretty informal. In propositional logic, an interpretation is an assignment of truth values to sentential variables. Now an interpretation's going to be something more complicated. An interpretation is made up of a set and three mappings.

Slide 5.2.2
The set is the universe, U, which is a set of objects. So what's an object? Well, really, it could be this chair and that chair and these pieces of chalk or it could be all of you guys or it could be some trees out there, or it could be rather more abstract objects like meetings or points in time or numbers. An object could be anything you can think of, and the universe can be any set (finite or infinite) of objects. The universe is also sometimes called the "domain of discourse."
Slide 5.2.3
There's a mapping from constant symbols to elements of U, specifying how names are connected to objects in the world. So I might have the constant symbol, Fred, and I might have a particular person in the universe, and then the interpretation of the symbol Fred could be that person.

Slide 5.2.4
The next mapping is from predicate symbols to relations on U. An n-ary relation is a set of lists of n objects, saying which groups of things stand in that particular relation to one another. A binary relation is a set of pairs. So if I have a binary relation "brother of" and U is a bunch of people, then the relation would be the set of all pairs of people such that the second is the brother of the first.

Slide 5.2.5
The last mapping is from function symbols to functions on U. Functions are a special kind of relation, in which, for any particular assignment of the first n-1 elements in each list, there is a single possible assignment of the last one. In the brother-of relation, there could be many pairs with the same first item and a different second item, but in a function, if you have the same first item then you have to have the same second item. So that means you just name the first item and then there's a unique thing that you get from applying the function. So it's OK for mother-of to be a function, discounting adoptions and other unusual situations. We will also, for now, assume that our functions are total, which means that there is an entry for every possible assignment of the first n-1 elements.

So, the last mapping is from function symbols to functions on the universe.

Slide 5.2.6
Before we can do the part of semantics that says what sentences are true in which interpretation, we have to talk about what terms mean. Terms name things, but we like to be fancy so we say a term denotes something, so we can talk about the denotation of a term, that is, the thing that a term names.
Slide 5.2.7
The denotations of constant symbols are given directly in the interpretation.

Slide 5.2.8
The denotation of a variable is undefined. What does x mean, if x is a variable? The answer is, "mu." It doesn't mean anything. That's a Zen joke. If you don't get it, don't worry about it.

Slide 5.2.9
The denotation of a complex term is defined recursively. So, to find the interpretation of a function symbol applied to some terms, first you look up the function symbol in the interpretation and get a function. (Remember that the function symbol is a syntactic thing, ink on paper, but the function it denotes is an abstract mathematical object.) Then you find the interpretations of the component terms, which will be objects in U. Finally, you apply the function to the objects, yielding an object in U. And that object is the denotation of the complex term.

Slide 5.2.10
In the context of propositional logic, we looked at the rules of semantics, which told us how to determine whether a sentence was true in an interpretation. Now, in first-order logic, we'll add some semantic rules, for the new kinds of sentences we've introduced. One of our new kinds of sentences is a predicate symbol applied to a bunch of terms. That's a sentence, which is going to have a truth value, true or false.
Slide 5.2.11
To figure out its truth value, we first use the denotation rules to find out which objects are named by each of the terms. Then, we look up the predicate symbol in the interpretation, which gives us a mathematical relation on U. Finally, we look to see if the list of objects named by the terms is a member of the relation. If so, the sentence is true in the given interpretation.

Slide 5.2.12
Let’s look at an example. Imagine we want to determine whether the sentence Brother(Jon, Joe) is true in some interpretation.

Slide 5.2.13
First, we look up the constant symbol “Jon” in the interpretation and find that it names this guy with glasses.

Slide 5.2.14
Then we look up “Joe” and find that it names this angry-looking guy.
Slide 5.2.15
Now we look up the predicate symbol "Brother" and find that it denotes this complicated relation.

Slide 5.2.16
Finally, we look up the pair of the guy with glasses and the angry-looking guy, to see if they're in the relation. They are, so the sentence must be true in that interpretation. It's easy to think of lots of other interpretations in which it wouldn't be true (and lots of others in which it would).

Slide 5.2.17
Another new kind of sentence we introduced has the form term1 = term2. The semantics are pretty unsurprising: if the object denoted by term1 is the same as the object denoted by term2, then the sentence holds.

Slide 5.2.18
It's important to note that two different constant symbols can denote the same object in the universe; so this is not a test on equality of names. We might have an interpretation that maps the symbols Jon and Jack both into the same guy. In that case, Jon equals Jack holds in I.
Now we have to figure out how to tell whether sentences with quantifiers in them are true.

**Slide 5.2.20**

In order to talk about quantifiers we need the idea of extending an interpretation. We would like to be able to extend an interpretation to bind variable $x$ to value $A$. We'll write that as $I$ with $x$ bound to $A$. Here, $x$ is a variable and $A$ is an object; an element of $U$. The idea is that, in order to understand whether a sentence that has variables in it is true or not, we have to make various temporary assignments to the variables and see what the truth value of the sentence is. Binding $x$ to $A$ is kind of like adding $x$ as a constant symbol to $I$. It's kind of like temporarily binding a variable in a programming language.

**Slide 5.2.21**

Now, how do we evaluate the truth under interpretation $I$, of the statement "for all $x$. $\Phi$"? So how do we know if that's true? Well, it's true if and only if $\Phi$ is true if for every possible binding of variable $X$ to thing in the world $A$. Okay? For every possible thing in the world that you could plug in for $X$, this statement's true. That's what it means to say "for all $x$. $\Phi$".

**Slide 5.2.22**

Similarly, to say that there exists an $x$ such that $\Phi$, it means that $\Phi$ has to be true for some $A$ in $U$. That is to say, there has to be something in the world such that if we plug that in for $x$, then $\Phi$ becomes true.
It's hard to understand the precedence of these operators using the usual rules. A quantifier is understood to apply to everything to its right in the formula, stopping only when it reaches an enclosing close parenthesis.

So in this example sentence, the for all x applies until the close paren after the first Q(x); and the exists x applies to the end of the sentence.

All right, let's work on an example. Here's a picture of our world.

There are four things in our U. Here they are.
We have one constant symbol, Fred.

We have four predicates: Above, Circle, Oval, Square. The numbers above them indicate their arity, or the number of arguments they take. Now these particular predicate names suggest a particular interpretation. The fact that I used this word, "Circle", makes you guess that probably the interpretation of circle is going to be true for the red object. But of course it needn't be. The fact that those marks on the page are like an English word that we think means something about the shape of an object, that doesn't matter. The syntax is just some words that we write down on our page. But it helps us understand what's going on. It's just like using reasonable variable names in a program that you might write. When you call a variable "the number of times I've been through this loop," that doesn't mean that the computer knows what that means. It's the same thing here.

And we have one function symbol, called "hat", that takes a single argument.

Now we can talk about a particular interpretation, I. We'll define I so that I(Fred) is the triangle.
Slide 5.2.31
Now, what kind of a thing is I(Above)? Well, Above is a predicate symbol, and the interpretation of a predicate symbol is a relation, so I(Above) is a relation. Here's the particular relation we define it to be: it's a set of pairs, because above has arity 2. It contains every pair of objects for which we want the relation "Above" to be true.

Slide 5.2.32
The interpretation of Circle is a unary relation. As you might expect in this world, it's the singleton set, whose element is a one-tuple containing the circle. (Of course, it doesn't have to be!).

Slide 5.2.33
We'll interpret the predicate Oval to be true of both the oval object and the round one (circles are a special case of ovals, after all).

Slide 5.2.34
And we'll say that the hat of the triangle is the square and the hat of the oval is the circle. If we stopped at this point, we would have a function, but it wouldn't be total (it wouldn't have an entry for every possible first argument). So, we'll make it total by saying that the square's hat is the square and the circle's hat is the circle.
Slide 5.2.35
Finally, just to cause trouble, we'll interpret the predicate "Square" to be true of the triangular object.

Slide 5.2.36
Now, let's find the truth values of some sentences in this interpretation. What about Square(Fred), is that true in this interpretation? Yes. We look to see that Fred denotes the triangle, and then we look for the triangle in the relation denoted by square, and we find it there. So the sentence is true.

Slide 5.2.37
What about this one? Is Fred above its hat?

Slide 5.2.38
Let's start by asking the question, what's the denotation of the term, hat(Fred)? It's the square, right? We look up Fred, and get the triangle. Then we look in the hat function, and, sure enough, there's a pair with triangle first and square second. So hat(Fred) is a square.
Slide 5.2.39
Now the question is: does the Above relation hold of the triangle and the square? We look this pair up in the relation denoted by Above, and we can’t find it. So the Above relation doesn’t hold of these objects.

Slide 5.2.40
And our original sentence is false.

Slide 5.2.41
Okay. What about this sentence: there exists an x such that Oval x. Is there a thing that is an oval? Yes. So how do we show that carefully?

Slide 5.2.42
We say that there's an extension of this interpretation where we take x and substitute in for it, the circle. Temporarily, I say that I(x) is a circle. And now I ask, in that new interpretation, is it true that Oval(x). So I look up x and I get the circle. I look up Oval and I get the relation with the circle and the oval, and so the answer's yes.
Here's a more complicated question in the same domain and interpretation. Is the sentence: “For all x there exists a y such that either x is Above y or y is Above x” true in I?

We can tell whether this is true by going through every possible object in the universe and binding it to the variable x, and then seeing whether the rest of the sentence is true. So, for example, we might put in the triangle for x, just to start with.

Now, having made that binding, we have to ask whether the sentence “There exists a y such that either x is Above y or y is Above x” is true in the new interpretation. Existentials are easier than universals; we just have to come up with one y that makes the sentence true. And we can; if we bind y to the square, then that makes Above(y,x) true, which makes the disjunction true. So, we've proved this existential statement is true.

If we can do that for every other binding of x, then the whole universal sentence is true. You can verify that it is, in fact, true, by finding the truth value of the sentence with the other objects substituted in for x.
Slide 5.2.47

Okay. Here's our last example in this domain. What about the sentence: "for all x, for all y, x is Above y or y is Above x"? Is it true in interpretation I?

Slide 5.2.48

If it's going to be true, then it has to be true for every possible instantiation of x and y to elements of U. So, what, in particular, about the case when x is the square and y is the circle?

Slide 5.2.49

We can't find either the pair (square, circle), or the pair (circle, square) in the above relation, so this statement isn't true.

And, therefore, neither is the universally quantified statement.
Slide 5.3.1
Now we're going to see how first-order logic can be used to formalize a variety of real-world concepts and situations. In this batch of problems, you should try to think of the answer before you go on to see it.

Slide 5.3.2
How would you use first-order logic to say "Cats are mammals"? (You can use a unary predicate "cat" and another unary predicate "mammal").

Slide 5.3.3
For all x Cat(x) implies Mammal(x). This is saying that every individual in the cat relation is also in the mammal relation. Or that cats are a subset of mammals.

Slide 5.3.4
All right. Let's let Jane be a constant. Tall and Surveyor can be unary predicates. How can we say Jane is a tall surveyor?
Slide 5.3.5
Surveyor of Jane and Tall of Jane.

Slide 5.3.6
A nephew is a sibling's son. Nephew, Sibling, and Son are all binary relations. I'll start you off and say for all x and y, x is the nephew of y if and only if something. In English, what relationship has to hold between x and y for x to be a nephew of y? There has to be another person z who is a sibling of y and x has to be the son of z.

Slide 5.3.7
So, the answer is, for all x and y, x is the Nephew of y if and only if there exists a z such that y is a Sibling of z and x is a Son of z.

Slide 5.3.8
When you have relationships that are functional, like mother-of, and maternal-grandmother-of, then you can write expressions using functions and equality. So, what's the logical way of saying that someone's maternal grandmother is their mother's mother? Use mgm, standing for maternal grandmother, and mother-of, each of which is a function of a single argument.
Slide 5.3.9
We can say that, for all $x$ and $y$, $x$ is the maternal grandmother of $y$ if and only if there exists a $z$ such that $x$ is the mother of $z$, and $z$ is the mother of $y$.

Slide 5.3.10
Using a binary predicate $\text{Loves}(x,y)$, how can you say that everybody loves somebody?

Slide 5.3.11
This one’s fun, because there are really two answers. The usual answer is for all $x$, there exists a $y$ such that $\text{Loves}(x,y)$. So, for each person, there is someone that they love. The loved-one can be different for each lover. The other interpretation is that there is a particular person that everybody loves. How would we say that?

Slide 5.3.12
There exists a $y$ such that for all $x$, $\text{Loves}(x,y)$. So, just by changing the order of the quantifiers, we get a very different meaning.
Slide 5.3.13
Let's say nobody loves Jane. Poor Jane. How can we say that?

Slide 5.3.14
For all x, not Loves(x, Jane). So, for everybody, every single person, that person doesn't love Jane.

Slide 5.3.15
An equivalent thing to write is there does not exist an x such that Loves(x, Jane). This is a general transformation, if you have for all x not something, then it's the same as having not there exists an x something. It's like saying I can't find a single x such that x Loves Jane.

Slide 5.3.16
Everybody has a father.
Slide 5.3.17
For all x, there exists y such that Father(y, x).

Slide 5.3.18
Everybody has a father and a mother.

Slide 5.3.19
For all x, there exists y and z such that Father(y, x) and Mother(z, x).

Slide 5.3.20
Now, you might ask whether y and z are necessarily different. The answer is, no, that's not enforced by the logic. For that matter, they could be the same as x. Now, if you use the typical definitions of father and mother, they won't be the same, but that's up to the interpretation.
Slide 5.3.21
Whoever has a father has a mother.

Slide 5.3.22
This is a general statement about objects of the kind, everything that has one property has another property. All right? So we’ll talk about everything by starting with forall x.

Slide 5.3.23
Now, how do we describe x’s that have a father? Exists y such that Father(y,x).

Slide 5.3.24
And we can describe x’s that have a mother by Exists y. Mother(y,x).
Slide 5.3.25
Finally, we put these together using implication, just as we did with the "all cats are mammals" example. We want to say objects with a Father are a subset (in this case, it will turn out they're a proper subset) of the set of objects with a Mother. So, we end up with for all x, if there exists a y such that y is the Father of x, then there exists a y such that y is the Mother of x.

Slide 5.3.26
Note that the two variables named y have separate scopes, and are entirely unrelated to one another. You could rename either or both of them and the semantics of the sentence would remain the same. It's technically legal to have nested quantifiers over the same variable, and there are rules for figuring out what it means, but it's very confusing, so it's just better not to do it.

Slide 5.4.1
Now that we understand something about first-order logic as a language, we'll talk about how we can use it to do things. As in propositional logic, the thing that we'll most often want to do with logical statements is to figure out what conclusions we can draw from a set of assumptions. In propositional logic, we had the notion of entailment: a KB entails a sentence iff the sentence is true in every interpretation that makes KB true.
In first-order logic, the notion of entailment is the same. A knowledge base entails a sentence iff the sentence holds in every interpretation in which the knowledge base holds.

It's important that entailment is a relationship between a set of sentences, KB, and another sentence, S. It doesn't directly involve a particular intended interpretation that we might have in mind. It has to do with the subsets of all possible interpretations in which KB and S hold; entailment requires that the set of interpretations in which KB holds be a subset of those in which S holds. This is sort of a hard thing to understand at first, since the number (and potential weirdness) of all possible interpretations in first-order logic is just huge.

In propositional logic, we were able to think about computing entailment by doing a brute-force enumeration of all interpretations, then, in each interpretation, checking to see whether the sentence and/or the knowledge base were true in that interpretation. This will be impossible in first-order logic for two reasons. First, it's completely out of the question to enumerate all possible interpretations. How many universes are there? More than I can count...

Second, even for a single interpretation, it's not necessarily possible to compute whether a sentence holds in that interpretation. Why? Because if it has a universal or existential quantifier, it will require testing whether a sentence holds for every substitution of an element in U for the quantified variable. And if the universe is infinite, you just can't do that.
Let's look at a particular situation in which we might want to do logical inference. Consider our shapes example from before. Let's say that we know, as our knowledge base, that all circles are ovals, and that no squares are ovals. We can write this as for all x circle of x implies oval of x. And for all x square of x implies not oval of x.

Now, let's say we're wondering whether it's also true that no squares are circles. We'll call that sentence s, and write it for all x square of x implies not circle of x.

We know KB holds in interpretation I, and we wonder whether S holds in I.

We could answer this by asking the question: Does KB entail S? Does our desired conclusion follow from our assumptions.
Slide 5.4.10

You might say that entailment is too big a hammer. I don't actually care whether S is true in all possible interpretations that satisfy KB. Why? because I have a particular interpretation in mind (namely, our little world of geometric shapes, embodied in interpretation I). And I know that KB holds in I. So what I really want to know is whether S holds in I.

Unfortunately, the computer does not know what interpretation I have in mind. We want the computer to be able to reach valid conclusions about my intended interpretation without my having to enumerate it (because it may be infinite).

For this particular example of I, it's not too hard to check whether S holds (because the universe is finite and small). But, as we said before, in general, we won't be able even to test whether a sentence holds in a particular interpretation.

Slide 5.4.11

Let's look at our KB for a minute. When we wrote it down, we had a particular interpretation in mind, as evidenced by the names of the propositions. But now, here's another interpretation, \( I_1 \):

\[
\begin{align*}
\text{KB: } & \forall x. \text{Circle}(x) \rightarrow \text{Oval}(x) \land \forall x. \text{Square}(x) \rightarrow \neg \text{Oval}(x) \\
S: & \forall x. \text{Square}(x) \rightarrow \neg \text{Oval}(x)
\end{align*}
\]

An Infinite Interpretation

\[
\begin{align*}
\text{KB: } & \forall x. \text{Circle}(x) \rightarrow \text{Oval}(x) \land \forall x. \text{Square}(x) \rightarrow \neg \text{Oval}(x) \\
S: & \forall x. \text{Square}(x) \rightarrow \neg \text{Oval}(x)
\end{align*}
\]

\[
U_1 = \{1, 2, 3, \ldots\}
\]

Slide 5.4.12

The universe is the positive integers (numbers 1, 2, etc.). This universe is clearly infinite.

Slide 5.4.13

Let's let the circle relation stand for positive integers evenly divisible by 4. So it's the infinite set \( \{4, 8, 12, 16, \ldots\} \).
We'll let oval stand for the even positive integers, \{2, 4, 6, 8, \ldots\}.

And we'll let square stand for the odd positive integers, \{1, 3, 5, 7, \ldots\}.

Now, does KB hold in I1?

We can't verify that by enumerating U and checking the sentences inside the universal quantifier. However, we all know, due to basic math knowledge, that it does.
Similarly, we can see that \( S \) holds in I1, as well. Unfortunately, we can't rely on our computers to be as smart as we are (yet!). So, if we want a computer to arrive at the conclusion that \( S \) follows from \( KB \), it will have to do it more mechanically.

Let's think about a different \( S \), which we'll call \( S_1 \): For every circle and every oval that is not a circle, the circle is above the oval.

Back in I, our original geometric interpretation, this sentence holds, right? But does it "follow from" \( KB \)? Is it entailed by \( KB \)?

No. We can see this by going back to interpretation I1, and letting the interpretation of the "above" relation be greater-than on integers.
### Slide 5.4.22

Then $S$ holds in $I_1$ if all integers divisible by 4 are greater than all integers divisible by 2 but not by 4, which is clearly false.

### Slide 5.4.23

So, although $KB$ and $S$ both hold in our original intended interpretation $I$, $KB$ does not entail $S$, because we can find an interpretation in which $KB$ holds but $S$ does not.

### Slide 5.4.24

We can see from this example that entailment captures this general notion of a sentence following from a set of assumptions; of being able to justify the truth of $S$ based only on the truth of $KB$.

### Slide 5.4.25

So, we like the notion of entailment, but we can’t compute it directly.
So what do we do? As we did in propositional logic, we will stay in the domain of syntax, and do proofs to figure out whether S is entailed by KB.

There are proof rules that are sound and complete, in the sense that if S is entailed by KB, there is a finite proof of that. So, it’s easier, in general, though not for every particular case, to do a proof of general entailment than to test whether a sentence holds in a given interpretation.

The next few segments of this material will show how to extend the notion of resolution refutation from propositional logic to first-order logic.

We just argued that entailment is the right notion when we want to ask the question whether a sentence S follows from a KB. And that we’re going to show entailment via proof.

But what if we have a particular interpretation in mind? We’ve seen that we can’t in general, test whether a sentence holds in that interpretation. How can we use the ability to use proof to show entailment, in order to test whether a sentence holds in an interpretation?

The answer is that we have to axiomatize our domain. That is, we have to write down a set of sentences, or axioms, which will serve as our KB.
Ideally the axioms would be so specific that there was a single interpretation, our intended interpretation, in which they held. In general, though, this will be impossible. You might be able to constrain your axioms to describe domains that contain exactly 4 objects, but you’ll never be able to say exactly which 4. You can often give axioms that put stringent enough requirements on the relationships between those objects that all of the interpretations in which the axioms hold are essentially the same as (isomorphic to) your intended interpretation.

No matter how constraining your axioms are, you can rely on the fact that if your KB holds in your intended interpretation and KB entails S, then S holds in the intended interpretation.

But that’s only half of what we need. There might be some fact, S, about your intended interpretation that you would like to be able to derive from your axioms. But, if your axioms are not specific enough, then they might admit some interpretations in which S does not hold, and in that case, the axioms will not entail S, even though it might hold in the intended interpretation.

Let’s work through an example of axiomatizing a domain. We’ll think about our good old geometric domain, but, to simplify matters a bit, let’s assume that we have the constant symbols A, B, C, and D. And let our interpretation specify that A is the square, B is the circle, C is the triangle, and D is the oval.
We propose to axiomatize this domain by specifying the above relation on these constants: Above(A,C) and Above(B,D).

And we'll give some axioms that say how the hat function can be derived from Above: for all x and y, if x is above y, then hat of y equals x; and for all x, if there is no y such that y is above x, then hat of x equals x.

These four axioms will constitute our KB. Now, we're curious to know whether it's okay to conclude that the hat of A is A. It's true in our intended interpretation, and we'd like it to be a consequence of our axioms.

So, does our KB entail S? Unfortunately not. Consider the interpretation I2. It has two extra pairs in the interpretation of Above. Our axioms definitely hold in this interpretation, but S does not. In fact, in this interpretation, the sentence hat(A) = C will hold.
Just so we can see what's going on, let's go back to our Venn diagram for entailment. In this case, the blue set of interpretations in which the KB holds is not a subset of the green set of interpretations in which S holds. So, it is possible to have an interpretation, I2, in which KB holds but not S. KB does not entail S (for it to do so, the blue area would have to be a subset of the green), and so we are not licensed to conclude S from KB.

How can we fix this problem? We need to add more axioms, in order to rule out I2 as a possible interpretation. (Our goal is to make the blue area smaller, until it becomes a subset of the green area).

Here's a reasonable axiom to add: for all x and y, if x is above y then y is not above x. It says that above is asymmetric. With this axiom added to our KB, KB no longer holds in I2, and so our immediate problem is solved.

But we're not out of the woods yet. Now consider interpretation I3, in which the circle is above the square. KB holds in I3, but S does not. So S is still not entailed by I3.

Clearly, we're missing some important information about our domain. Let's add the following important piece of information to our set of axioms: there is nothing above the square or the circle.
Slide 5.4.42
If we let our new KB have these axioms as well, then it fails in I3, and does, in fact, entail S. Whew.

Slide 5.4.43
So, when you are axiomatizing a domain, it’s important to be as specific as you can. You need to find a way to say everything that’s crucial about your domain. You will never be able to draw false conclusions, but if you are too vague, you may not be able to draw some of the conclusions that you desire.

It turns out, in fact, that there is no way to axiomatize the natural numbers without including some weird unintended interpretations that have multiple copies of the natural numbers.

Still this shouldn’t deter us from the enterprise of using logic to formalize reasoning inside computers. We don’t have any substantially better alternatives, and, with care, we can make logic serve a useful purpose.

6.034 Notes: Section 5.5

Slide 5.5.1
We are going to use resolution refutation to do proofs in first-order logic. It’s a fair amount trickier than in propositional logic, though, because now we have variables to contend with.
Let's try to get some intuition through an example. Imagine you knew "for all X, P of X implies Q of X." And let's say you also knew P of A. What would you be able to conclude? Q of A, right? You ought to be able to conclude Q of A.

This is actually Aristotle's original syllogism: From "All men are mortal" and "Socrates is a man", conclude "Socrates is a mortal".

So, how can we justify this conclusion formally? Well, the first step would be to get rid of the implication.

Next, we could substitute the constant A in for the variable x in the universally quantified sentence. By the semantics of universal quantification, that's allowed. A universally quantified statement has to be true of every object in the universe, including whatever object is denoted by the constant symbol A. And now, we can apply the propositional resolution rule.

The hard part is figuring out how to instantiate the variables in the universal statements. In this problem, it was clear that A was the relevant individual. But it not necessarily clear at all how to do that automatically.
Now, we have to do two jobs before we can see how to do first-order resolution. The first is to figure out how to convert from sentences with the whole rich structure of quantifiers into a form that lets us use resolution. We'll need to convert to clausal form, which is a kind of generalization of CNF to first-order logic.

The second is to automatically determine which variables to substitute in for which other ones when we're performing first-order resolution. This process is called unification.

We'll do clausal form next, then unification, and finally put it all together.

Clausal form (which is also sometimes called "prenex normal form") is like CNF in its outer structure (a conjunction of disjunctions, or an "and" of "ors"). But it has no quantifiers. Here's an example conversion.

**Converting to Clausal Form**

The first step you guys know very well is to eliminate arrows. You already know how to do that. You convert an equivalence into two implications. And anywhere you see alpha right arrow beta, you just change it into not alpha or beta.
Slide 5.5.10
The next thing you do is drive in negation. You already basically know how to do that. We have deMorgan's laws to deal with conjunction and disjunction, and we can eliminate double negations.

As a kind of extension of deMorgan's laws, we also have that not for all x alpha turns into exists x not alpha. And that not exists x alpha turns into for all x not alpha. The reason these are extensions of deMorgan's laws, in a sense, is that a universal quantifier can be seen abstractly as a conjunction over all possible assignments of x, and an existential as a disjunction.

Slide 5.5.11
The next step is to rename variables apart. The idea here is that every quantifier in your sentence should be over a different variable. So, if you had two different quantifications over x, you should rename one of them to use a different variable (which doesn't change the semantics at all). In this example, we have two quantifications involving the variable x. It's especially confusing in this case, because they're nested. The rules are like those for a programming language: a variable is captured by the nearest enclosing quantifier. So the x in Q(x,y) is really a different variable from the x in P(x). To make this distinction clear, and to automate the downstream processing into clausal form, we'll just rename each of the variables.

Slide 5.5.12
Now, here's the step that many people find confusing. The name is already a good one. Step four is to skolemize, named after a logician called Thoralf Skolem. Imagine that you have a sentence that looks like: there exists an X such that P of X. The goal here is to somehow arrive at a representation that doesn't have any quantifiers in it. Now, if we only had one kind of quantifier, it would be easy because we could just mention variables and all the variables would be implicitly quantified by the kind of quantifier that we have. But because we have two quantifiers, if we dropped all the quantifiers off, there's a mess, because you don't know which kind of quantification is supposed to apply to which variable.

Slide 5.5.13
The Skolem insight is that when you have an existential quantification like this, you're saying there is such a thing as a unicorn, let's say that P means "unicorn". There exists a thing such that it's a unicorn. You can just say, all right, well, if there is one, let's call it Fred. That's it. That's what Skolemization is. So instead of writing exists an X such that P of X, you say P of Fred. The trick is that it absolutely must be a new name. It can't be any other name of any other thing that you know about. If you're in the process of inferring things about John and Mary, then it's not good to say, oh, there's a unicorn and it's John -- because that's adding some information to the picture. So to Skolemize, in the simple case, means to substitute a brand-new name for each existentially quantified variable.
Slide 5.5.14
For example, if I have exists X and Y such that R of XY, then it's going to have to turn into R of Thing1, Thing2. Because we have two different variables here, they have to be given different names.

Slide 5.5.15
But the names also have to persist so that if you have exists an X such that P of X and Q of X, then if you skolemize that expression you should get P of Fleep and Q of Fleep. You make up a name and you put it in there, but every occurrence of this variable has to get mapped into that same unique name.

Slide 5.5.16
If you have different quantifiers, then you need to use different names.

Slide 5.5.17
All right. If that's all we had to do it wouldn't be too bad. But there's one more case. We can illustrate it by looking at two interpretations of "Everyone loves someone". In the first case, there is a single y that everyone loves. So we do ordinary skolemization and decide to call that person Englebert.
In the second case, there is a different \( y \), potentially, for each \( x \). So, if we were just to substitute in a single constant name for \( y \), we’d lose that information. We’d get the same result as above, which would be wrong. So, when you are skolemizing an existential variable, you have to look at the other quantifiers that contain the one you’re skolemizing, and instead of substituting in a new constant, you substitute in a brand new function symbol, applied to any variables that are universally quantified in an outer scope.

In this case, what that means is that you substitute in some function of \( x \), for \( y \). Let’s call it beloved of \( x \). Now it’s clear that the person who is loved by \( x \) depends on the particular \( x \) you’re talking about.

So, in this example, we see that the existential variable \( w \) is contained in the scope of two universally quantified variables, \( x \) and \( z \). So, we replace it with \( G(x,z) \), which allows it to depend on the choices of \( x \) and \( z \).

Note also, that I’ve been using silly names for Skolem constants and functions (like Englebert and Beloved). But you, or the computer, are only obliged to use new ones, so things like \( F123221 \) are completely appropriate, as well.

Now we can drop the universal quantifiers because we just replaced all of the existential quantifiers with Skolem constants or functions. Now there’s only one kind of quantifier left, so we can just drop them without losing information.
And then we convert to clauses. This just means multiplying out the and's and the or's, because we already eliminated the arrows and pushed in the negations. We'll return a set of sets of literals. A literal, in this case, is a predicate applied to some variables and constants, or the negation of a predicate applied to some variables and constants.

I'm using set notation here for clauses, just to emphasize that they aren't lists; that the order of the literals within a clause and the order of the clauses within a set of clauses, doesn't have any effect on its meaning.

Finally, we can rename the variables in each clause. It's okay to do that because forall x P(x) and Q(x) is equivalent to forall y P(y) and forall z P(z). In fact, you don't really need to do this step, because we're assuming that you're always going to rename the variables before you do a resolution step.

So, let's do an example, starting with English sentences, writing them down in first-order logic, and converting them to clausal form. Later, we'll do a resolution proof using these clauses.

John owns a dog. We can write that in first-order logic as "there exists an x such that D(x) and O(J, x)". So, we're letting D stand for is-a-dog and O stand for owns and J stand for John.
Slide 5.5.26
To convert this to clausal form, we can start at step 4, Skolemization, because the previous three steps are unnecessary for this sentence. Since we just have an existential quantifier over x, without any enclosing universal quantifiers, we can simply pick a new name and substitute it in for x. Let's call x "Fido". This will give us two clauses with no variables, and we're done.

Example: Converting to clausal form

- a. John owns a dog
  \[ \exists x. D(x) \land O(x, y) \]
  \[ D(Fido) \land O(1, Fido) \]

- b. Anyone who owns a dog is a lover of animals
  \[ \forall x. (\exists y. D(y) \land O(x, y)) \rightarrow L(x) \]

Slide 5.5.27
Anyone who owns a dog is a lover of animals. We can write that in FOL as "For all x, if there exists a y such that D(y) and O(x, y), then L(x)." We've added a new predicate symbol L to stand for "is a lover of animals".

Example: Converting to clausal form

- a. John owns a dog
  \[ \exists x. D(x) \land O(x, y) \]
  \[ D(Fido) \land O(1, Fido) \]

- b. Anyone who owns a dog is a lover of animals
  \[ \forall x. (\exists y. D(y) \land O(x, y)) \rightarrow L(x) \]

Slide 5.5.28
First, we get rid of the arrow. Note that the parentheses are such that the existential quantifier is part of the antecedent, but the universal quantifier is not. The answer would come out very differently if those parens weren't there; this is a place where it's easy to make mistakes.

Example: Converting to clausal form

- a. John owns a dog
  \[ \exists x. D(x) \land O(x, y) \]
  \[ D(Fido) \land O(1, Fido) \]

- b. Anyone who owns a dog is a lover of animals
  \[ \forall x. (\exists y. D(y) \land O(x, y)) \rightarrow L(x) \]

Slide 5.5.29
Next, we drive in the negations. We'll do it in two steps. I find that whenever I try to be clever and skip steps, I do something wrong.
Slide 5.5.30
There’s no skolemization to do, since there aren’t any existential quantifiers. So, we can just drop the universal quantifiers, and we’re left with a single clause.

Slide 5.5.31
Lovers of animals do not kill animals. We can write that in FOL as "For all x, if L(x) then for all y, A(y) implies not K(x,y)". We’ve added the predicate symbol A to stand for “is an animal” and the predicate symbol K to stand for x kills y.

Slide 5.5.32
First, we get rid of the arrows, in two steps.

Slide 5.5.33
Then we’re left with only universal quantifiers, which we drop, yielding one clause.
Slide 5.5.34
We just have three more easy ones. "Either Jack killed Tuna or curiosity killed Tuna." Everything here is a constant, so we get K(J,T) or K(C,T).

Slide 5.5.35
"Tuna is a cat" just turns into C(T).

Slide 5.5.36
And "All cats are animals" is not C(x) or A(x). I left out the steps here, but I'm sure you can fill them in.
Okay. Next, we'll see how to match up literals that have variables in them, and move on to resolution.