This section is not recommended for first-years.

Do they know what a free variable is?

Say something about how the $'$ operation on variables has no semantics.

Also, the cute facts about words beginning with 'sn' or with 'gl' will not be captured by the kind of semantics we are about to do.

Semantics is the process of allocating extensions to intensions.

0.1 Syntactic types for the various pieces of syntax

Constants and individual variables are of type `ind`
One-place predicate symbols are of type `ind -> bool`
One-place function symbols are of type `ind -> ind`
Quantifiers are of type `(ind -> bool) -> bool`
Determiners are of type `(ind -> bool) -> ((ind -> bool) -> bool)`

Semantics is useful for independence results. Peirce's law independent of K and S. Mind you, to do that we have to un-reserve $\to$.

1 Truth and satisfaction

In this section we develop the ideas of truth and validity (which we first saw in the case of propositional logic) in the rather more complex setting of predicate logic.

We are going to say what it is for a formula to be **true** in a structure. We will achieve this by doing something rather more general. What we will give is–for each language $\mathcal{L}$–a definition of what it is for a formula of $\mathcal{L}$ to be true in a structure. Semantics is a relation not so much between an expression and a structure as between a language and a structure.

We know what expressions are, so what is a structure? It's a set with knobs on. You needn't be alarmed here by the sudden appearance of the word 'set'. You don't need to know any fancy set theory to understand what is going on. The set in question is called the *carrier set*, or *domain*. One custom in mathematics is to denote structures with characters in uppercase $\mathfrak{FRAKTUR}$ font, typically with an '$\mathfrak{M}$'.

The obvious examples of structures arise in mathematics and can be misleading and in any case are not really suitable for our expository purposes here. We can start off with the idea that a structure is a set-with-knobs on. Here is a simple example that cannot mislead anyone.

The carrier set is the set {Beethoven, Handel, Domenico Scarlatti} and the knobs are (well, is rather than are because there is only one knob in this case) the binary relation is the favourite composer of. We would obtain a different structure by adding a second relation: *is-older-than* perhaps.

If we are to make sense of the idea of an expression being true in a structure then the structure must have things in it to match the various gadgets in the language to which the expression belongs. If the expression contains a two-place relation symbol 'loves' then the structure must have a binary relation on it to correspond. This information is laid down in the **signature**. The signature of the structure in the composers example above has one binary relation symbol and three constant symbols; the signature of set theory is equality plus one binary predicate; the signature of the language of first-order Peano arithmetic has slots for one unary function symbol, one nullary function symbol (or constant) and equality.

Let's have some illustrations, at least situations where the idea of a signature is useful.

- Cricket and baseball resemble each other in a way that cricket and tennis do not. One might say that cricket and baseball have the same signature. Well, more or less! They can be described by giving different values to the same set of parameters.

- It has been said that a French farce is a play with four characters, two doors and one bed. This *aperçu* is best expressed by using the concept of signature.

- Perhaps when you were little you bought mail-order kitsets that you assembled into things. When your mail-order kitset arrives, somewhere buried in the polystyrene chips you have a piece of paper (the "manifest") that tells you how many objects you have of each kind, but it does not tell you what to do with them. Loosely, the manifest is the *signature* in this example. Instructions on what you do with the objects come with the *axioms* (instructions for assembly).

- Recipes correspond to theories: lists of ingredients to signatures.

Structure	**Signature**	**Axioms**
French Farce	4 chars, 2 doors 1 bed	Plot
Dish	Ingredients	recipe
Kitset	list of contents	Instructions for assembly
cricket/baseball	innings, catches, etc	rules

1.1 Definition of "truth-in-a-structure"

First we need to decide what our structure is to be. Suppose it is $\mathfrak{M}$, with carrier set M. Next we need the concept of an **interpretation**. This is a function assigning to each predicate letter, function letter and constant in the language a subset of M^n, or a function $M^k \to M$, or element of M *mutatis mutandis*. That is to say, to each syntactic device in the language, the interpretation assigns a component of $\mathfrak{M}$ of the appropriate arity. For this to be possible it is necessary that the structure and our language have the same signature.

Now we have to get straight the difference between the rôles played by the various different kinds of lexical items ... what one might call the different parts of speech.

The first difference is between the logical vocabulary and the non-logical vocabulary. The non-logical vocabulary is open to various kinds of stipulation, whereas the nonlogical vocabulary isn't. You might object that the symbol '$\wedge$' acquires its meaning (logical conjunction) by stipulation, which of course it does. However, in this game, that particular kind of stipulation happened at the beginning of time, and it is not for us to choose to make '$\wedge$' mean anything else. Similarly '$\forall$', '$\exists$' and even '$=$'.

We are free to stipulate the meanings of the predicate symbols, the function letters, the constants and the variables. There are two ways in which we stipulate these, and it is helpful at this point to draw a contrast with the propositional case. The only nonlogical vocabulary in the propositional case is the propositional letters, and we stipulate them by means of **valuation**s. The meaning of the items of the logical vocabulary emerges from the way in which the semantics of complex expressions (complexed from the propositional letters) emerges from the semantics of those propositional letters. (I don't want to say *meaning* of the propositional letters co's in this case I mean *truth-value* rather than anything intensional such as meaning.)

In the propositional case we stipulate (by means of valuations) the semantics of the propositional letters, so that then the compositional rules for the connectives (the bits of logical vocabulary) tell us the truth value of the complex formulæ.

In the first-order case there are two kinds of non-logical vocabulary, and they are controlled by stipulation in two different ways. The two kinds are (i) constant symbols, predicate letters and function symbols and (ii) individual variables. Now this difference between the two kinds of non-logical vocabulary doesn't corrrespond neatly to a type distinction such as we have just seen. Individual variables will be given a semantics differently from constants, even tho' both these gadgets are of type `ind`. Constants are of a different syntactic type from predicate symbols, but their meanings are stipulated in the same way. . . in a way which we must now explain. reserved word

There is a difference in Compsci-speak between *configure time* and *run time*. If these expressions have any meaning for you, savour it. If not, don't worry.

It's at configure-time that we decide on the semantics of the constant symbols, predicate letters and function symbols. That's when we decide what the structure is to be. Suppose we fasten on a structure $\mathfrak{M}$. This act determines what the universe (what the totality of things over which our variables range) is going to be: it's going to be the carrier set M of $\mathfrak{M}$. The interpretation now tells us which element of M corresponds to which constant symbol, which subset of M corresponds to which one-place predicate letter, which subset of $M \times M$ corresponds to which two-place predicate letter, which subset of M^n corresponds to which n-place predicate letter, which function $M^n \to M$ corresponds to which n-place function letter. Thus an interpretation is a function that sends

Constant symbols	to elements of M;
n-ary predicate symbols	to subsets of M^n;

n-adic function symbols	to functions $M^n \to M$;
Propositional constant symbols	to `true` or `false`

It matches up the signature-in-the-language with the signature in the structure.

I shall use the calligraphic letter '$\mathcal{I}$' to vary over interpretations.

We have now equipped the language with an interpretation so we know what the symbols mean, but not what the values of the variables are. In other words, settling on an interpretation has enabled us to reach the position from which we started when doing propositional logic.

When we did semantics for propositional logic I encouraged you to think of valuations (rows in the truth-table) as *states*, as *scenarios*. The idea will be useful here too. The interpretation tells us what the constant symbols point to, but it can't tell us what the variables point to. To understand how variables point to things we need a notion of state very redolent of the notion of state suggested by propositional valuations.

It's rather like the position we are in when contemplating a computer program but not yet running it. When we run it we have a concept of instantaneous state of the program: these states (snapshots) are allocations of values to the program variables. Let us formalise a concept of state.

An **assignment function** is a function that takes a variable and returns a member of M. (If we want to make it clear that we are talking about assignment functions under an interpretation $\mathcal{I}$ we will speak of $\mathcal{I}$-assignment functions.) What do we do with assignment functions? Reflect that we don't really want to say of a formula with a free variable in it that it is *true* or that it is *false*—not straightforwardly anyway. But we do want to say something of that nature that takes the assignment functions into account. The gadget we need is a relation-between-assignment-functions-and-formulæ of *satisfaction*. We want to be able to say that an assignment function **satisfies** a formula. Thus we say—for example—that an assignment function f satisfies the formula '$R(x)$' as long as that element of the domain M to which f sends the variable 'x' belongs to the subset of M to which the interpretation $\mathcal{I}$ sent the letter 'R'.

The satisfaction relation between formulæ and assignment functions is defined "compositionally"—by recursion on the subformula relation. Clearly an assignment function will satisfy a conjunction iff it satisfies both conjuncts and will satisfy a disjunction iff it satisfies at least one of the disjuncts. The recursion steps for the quantifiers are fiddly, and depend on whether or not we insist that the assignment functions be total.

1.1.1 The recursions for the quantifiers

This is annoying and fiddly, because what one has to do depends very sensitively on whether one's assignment functions are total or partial.

- If our assignment functions are total, one says that
  - f satisfies $(\exists x)(\phi(x))$ as long as f satisfies $\phi(x)$;

  - f satisfies $(\forall x)(\phi(x))$ as long as $\phi(x)$ is satisfied by every f' which differs from f only for input 'x'.

- If our assignments functions are allowed to be partial one says that
  - f satisfies $(\exists x)(\phi(x))$ as long as *either* (i) f is defined on 'x' and satisfies $\phi(x)$, *or* (ii) f is not defined on 'x' but has an extension $f' \supset f$ which satisfies $\phi(x)$;
  - f satisfies $(\forall x)(\phi(x))$ as long as $\phi(x)$ is satisfied by every f' which differs from f only for input 'x'.

I cannot put my hand on my heart and swear that I have got these right. Fiddly details like these one tends to disregard!! If you ever find that you absolutely have to be on top of this detail contact me and we'll go over it together.

Now we are in a position to define what it is for an expression ϕ to be **true-according-to-the-interpretation** $\mathcal{I}$. Again, we have two ways to proceed: (i) with total assignment functions (ii) with partial assignment functions.

1. If our assignment functions have to be total then ϕ is **true-according-to-the-interpretation** $\mathcal{I}$ iff every $\mathcal{I}$–assignment-function satisfies ϕ.

2. If our assignment functions may be partial then ϕ is **true-according-to-the-interpretation** $\mathcal{I}$ iff the empty assignment function satisfies ϕ.

1.2 An illustration

Let's illustrate with an example

$$(\exists x)(F(x) \wedge \neg(x = a))$$

The '$\exists$', the '$\wedge$' and the '$=$' are part of the logical vocabulary. No stipulation. The 'F' is a part of the nonlogical vocabulary and when we stipulate our domain M we also stipulate which subset of M is to be the extension of 'F'. 'a' too is part of the nonlogical vocabulary and when we stipulate our domain M we also stipulate which element of M is to be the extension/denotation of 'a'. The variable 'x' is a different matter. A valuation f satisfies '$F(x) \wedge \neg(x = a)$' if the member of M to which it sends the variable 'x' is something other than the denotation (according to the interpretation $\mathcal{I}$) of the constant symbol 'a' and moreover is a member of that subset of M to which the interpretation $\mathcal{I}$ has sent the predicate letter 'F'. So is $(\exists x)(F(x) \wedge \neg(x = a))$ true? It is true as long as every valuation satisfies it. (Or if the empty valuation satisfies it—if our valuations are allowed to be partial)

In search of another example, we can return to Handel, Scarlatti and Beethoven. The language $\mathcal{L}$ has the signature of one binary relation "is the favourite composer of". The structure into which we are going to interpret $\mathcal{L}$ has carrier set

$$\{\text{Beethoven, Handel, Domenico Scarlatti}\}$$

and our interpretation sends "is the favourite composer of" to the set

$$\{\langle\text{Handel, Beethoven}\rangle, \langle\text{Handel, Domenico Scarlatti}\rangle, \langle\text{Domenico Scarlatti, Handel}\rangle\}.$$

then do some game-theoretic semantics!!

(warning: some people think this is meretricious)