Philosophy 1102: Introduction to Logic

Department of Philosophy
Langara College

Names, Predicates, Identity and Functions

1. Names and Predicates

The term 'atomic sentence' applies only to sentences of FOL (First Order Logic). One cannot really talk about an English sentence being atomic. But here's an English sentence that is easily translated into an atomic sentence of FOL:

> John Donne was a poet.

The first thing to note about this sentence is that the two main parts, 'John Donne' and 'poet' have very different kinds of meaning. For the meaning of 'John Donne' is a particular *thing*, a certain man who lived in England in the 17th century. The word 'poet', on the other hand, has no such meaning, for there have been many poets. (You might point at John Donne, but you cannot point at poet.)

Logicians *analyze* sentences, which means that they break them down into smaller parts. In analyzing a sentence, the first thing to do is identify all the *names*, or *individual constants*, that it contains. An individual constant is a word, or sequence of words, whose meaning is some single, particular object, such as John Donne. Thus the following are all names, or individual constants:

> Calgary
> Mars
> The Lions Gate Bridge
> Larry Campbell

Note that some objects in the world have no individual constant (in English, at least), and others have more than one name. The same is true of FOL, which can have multiple names for one object and no name for another. One difference between FOL and English is that FOL cannot have any names (like 'Zeus') that have no objective meaning, i.e. which don't refer to anything real.

Once all the names in a sentence have been identified, the analysis proceeds by removing it (or them) from the sentence, leaving a hole (or holes) behind. Thus the above sentence becomes:

> …… was a poet.

This sentence with a hole in it is called a *predicate*. In this case it's a one-place predicate, as it has just one hole.

Rather than using a row of dots to mark the hole, however, we use a 'variable', which is one of the (lower-case) letters t, u, v, w, x, y or z. (Usually we start with 'x' for the first hole, and then use 'y' and 'z' next.) So the predicate can be written:

x was a poet.

In FOL, this predicate would be written as something like:

Poet(x).

There are a few things to notice here.

1. A predicate in FOL is always a single word, in the sense that there are no spaces in it.

2. A predicate in FOL always begins with a capital letter.

3. The variable(s) that mark the hole(s) in the predicate are placed inside brackets that follow the predicate.

How can John Donne be referred to in FOL? We can define a name in FOL that refers to him, such as donne. Note that:

1. A name (individual constant) in FOL is a single word (no spaces).

2. A name always consists entirely of lower-case letters.

The FOL sentence saying that John Donne was a poet then becomes:

Poet(donne).

Note that, like some poetry, FOL is written backwards. It's like saying "A poet was John Donne". It may also seem odd that predicates are capitalized and names are not, as this is the opposite of what's done in English.

You may be wondering what happened to the *tense* of the English sentence. There are no tenses in FOL! FOL always speaks from a timeless perspective. The closest you can get to saying that John Donne *was* a poet is to say that John Donne *is* a poet in the 17th century. This might be written:

Poet(donne, century17)

Note that we're now treating the 17th century as an *object*, which is named by the individual constant 'century17'.

Two-Place Predicates

Consider the sentence: 'John is a client of Neil'. Here we have two proper names, 'John' and 'Neil', which may write as 'john' and 'neil' respectively in FOL. The predicate

'..... is a client of ----'

has two *places* (holes) in it, so it is called a *two-place predicate.* It is still written as a single word, beginning with a capital letter, say 'Client(x, y)'. We then translate the sentence as:

Client(john, neil).

Note that the order matters. 'Client(neil, john)' would say that Neil is a client of John, which is a different statement. We say that a two-place predicate expresses a *relation*.

An FOL sentence such as Client(john, neil), that involves a two-place predicate, is also called an atomic sentence.

Definition An atomic sentence is one that consists of a predicate, all of whose holes are filled in with names. It doesn't matter how many holes (or names) there are.

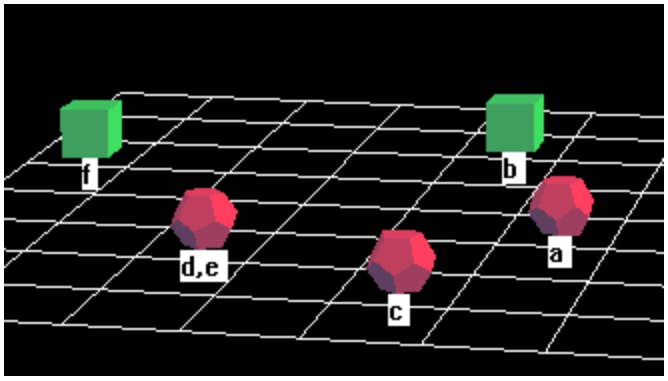
Exercise

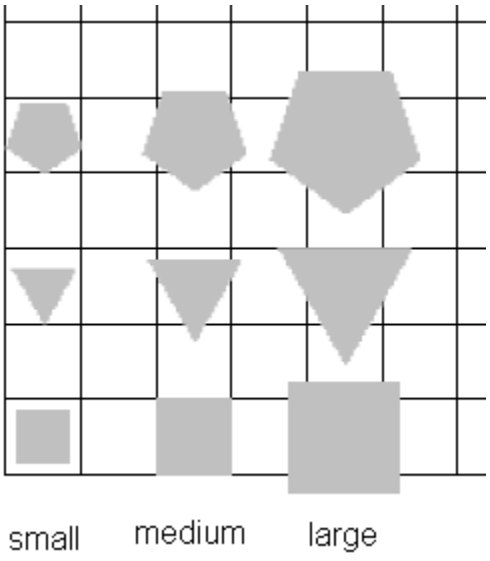
Identify all the individual constants in the following sentences and then specify all the one- or more-place predicates that can be obtained from each sentence by deleting one or more names.

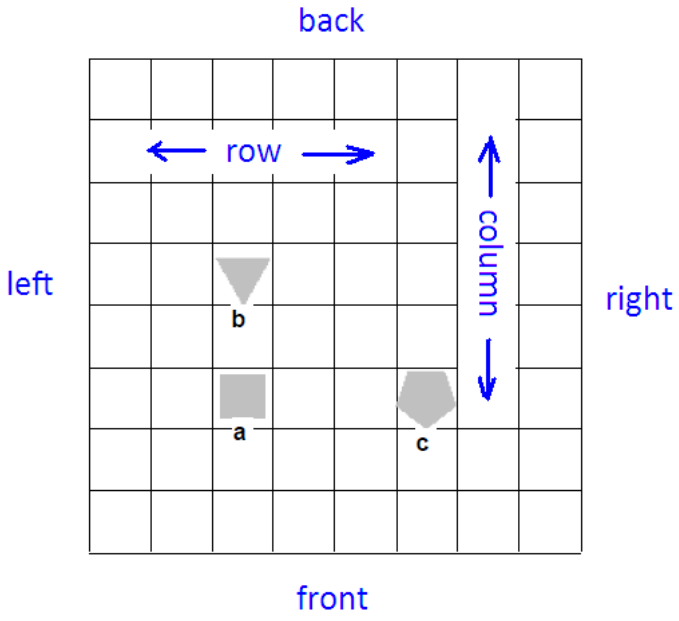
1. The Prime Minister is a Liberal.
2. The Liberal Party is currently in power.
3. Sarah attends UBC.
4. Charles is Sarah's room-mate.
5. Charles thinks his term paper is worth an A.
6. Fred's dentist wouldn't hurt a fly.
7. The Prime Minister assures us that his government can unite Canada.

For convenience, our textbook largely uses the same few predicates throughout, which are based on a game where solid figures, or blocks, are arranged on a grid. A particular arrangement of blocks is called a *world*. Two worlds, for illustration, are shown below.

Note that, by convention, the *front of the grid is drawn at the bottom*. There are three different shapes of block, namely cubes, tetrahedra and dodecahedra. To make drawing them easier, I use the '2D view' in which cubes are shown as squares, tetrahedra as triangles, and dodecahedra as pentagons (or even circles). Some objects can have ine or more names written beneath. Each of the three shapes also comes in three different sizes, called 'small', 'medium' and 'large', as shown on the right. Note that large objects spill over the edges of the cell, medium objects roughly fill the cell, and small objects fail to fill the cell.)







To describe these worlds, the book uses the following predicates (also listed on p. 22)

Tet(a)	a is a tetrahedron
Cube(a)	a is a cube
Dodec(a)	a is a dodecahedron
Small(a)	a is small
Medium(a)	a is medium
Large(a)	a is large
SameSize(a, b)	a is the same size as b
SameShape(a, b)	a is the same shape as b
Larger(a, b)	a is larger than b
Smaller(a, b)	a is smaller than b
SameCol(a, b)	a is in the same column as b
SameRow(a, b)	a is in the same row as b
Adjoins(a, b)	a and b are located on adjacent (but not diagonally) squares
LeftOf(a, b)	a is located nearer to the left edge of the grid than b
RightOf(a, b)	a is located nearer to the right edge of the grid than b
FrontOf(a, b)	a is located nearer to the front of the grid than b
BackOf(a, b)	a is located nearer to the back of the grid than b
Between(a, b, c)	a, b and c are in the same row, column, or diagonal, and a is between b and c

2. Atomic and Compound Sentences

An atomic sentence of FOL is just a predicate whose 'holes' are all filled by names. So every sentence in the left-hand column of the table above is atomic. We can regard an English sentence as atomic *when it has no parts that are themselves sentences*. Consider, for example, the sentence:

If Fred is lazy then he might fail.

There are two smaller sentences within this sentence, namely "Fred is lazy" and "He might fail". These are considered (declarative) sentences by logicians since they can be used to express a belief, and they can be either true of false. By contrast, the sentence:

A triathlete, after every race, smokes a cigarette, a cigar and then a pipe.

has no sentences within it. There is no part of this sentence that can be meaningfully asserted, all by itself.

3. Identity

Identity is a special relation, since it occurs in every academic subject, from astronomy to zoology, and indeed in everyday life. For this reason it is the one predicate that has a special symbol in FOL, with a fixed meaning. If object *c* is identical to object *d*, then we express this in FOL as:

$c = d$

What is identity, however? Philosophers take different views on this question, but here I shall present the standard view, expressed by Gottlob Frege in "On Sense and Meaning".

Identity is a relation that holds between each thing and *itself*. In other words, two objects are identical just in case they are not two objects at all, but actually one and the same object! Some well known identities are that the Morning Star is the Evening Star (a.k.a. the planet Venus) that Samuel Clemens is Mark Twain, and that Superman is Clark Kent (in the story at least). This relation of identity is sometimes called *numerical* identity by philosophers, for reasons that aren't clear to me. Perhaps it is because one can define identity as follows:

$c = d$ just in case the set $\{c, d\}$ has exactly *one* member.

$\{c, d\}$ is the set whose only members are *c* and *d*, by the way. If this set has only one member, then clearly *c* and *d* are *one* and the same object. (One is of course a number – is that the origin of the term?)

We see that the logical use of 'identity' is different from one common usage, for "identical twins" are certainly not numerically identical! (If they were identical, there would only be one person in the womb, and hence no twins.) In common usage, 'identical' often expresses a relation that philosophers call *exact similarity*. Two peas in a pod, for example, are exactly similar (well almost).

In English, identity is usually expressed using the word 'is'. For example, in the sentence:

The Morning Star *is* the Evening Star

the identity of the "two" planets is expressed by *is*. This is the shortest and simplest way to express identity. There are other ways, however, such as:

The Morning Star and the Evening Star are one and the same.

The Morning Star is the very same thing as the Evening Star.

and so on. You should be aware that the word *is* doesn't always express identity, however. For example, the sentence:

Fred is a dentist

is not an identity, as Fred isn't the same object as dentist. The word 'dentist' doesn't mean an object, but a *property* that some objects have. There is only one Fred, but many dentists. In this sentence, *is* is just a filler word, that connects 'Fred' to the predicate 'a dentist'. Some languages don't even have such a word.

Exercise

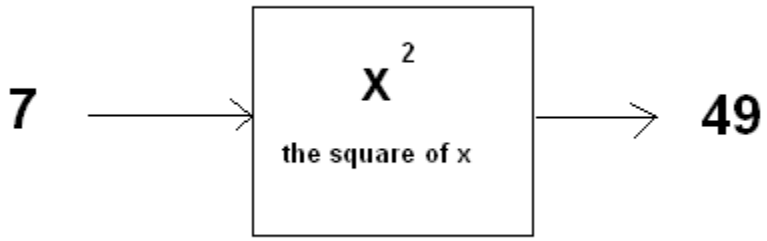
In the following sentences, write the '=' symbol under each 'is' that expresses identity. (N.B. Do not worry about whether the sentences are true of false.)

1. Janet is Fred's mother.
2. The capital of Alberta is Calgary.
3. Alice's swim coach is a woman.
4. Buying a house is a sound investment.
5. My favourite city is the one I was born in.

4. Functions

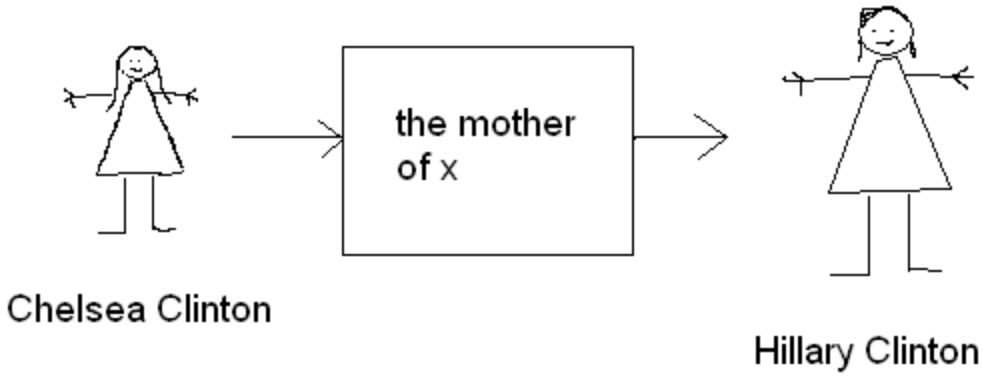
Underline all the names (singular terms, individual constants) from the previous exercise. You will see that some sentences contain *names within names*. For example, 'Fred's mother' is a name, and it has the name 'Fred' inside it. The logician Frege understood such compound names as expressing a *function*.

A function is technically a special kind of relation, one where each object in the domain is related to (or "maps to") exactly one object in the codomain. Some of you will be familiar with those mathematical terms. The basic idea is very simple, however, as one can think of a function as a kind of *machine* that accepts inputs, and produces a single output for each input. Consider, for example, the well-known *square* function:



This function accepts *numbers* as inputs (its "domain" is the set of numbers) and produces numbers as outputs. For a given input, it always produces the same output. For example, the input 7 always generates the output 49, and the input 3 always generates the output 9.

Not all functions operate on numbers. They can operate on any kind of object at all. Consider, for example, the *mother function*.



This function accepts *human beings* as inputs (not their names, the *actual person*). When you put someone into the machine's input, out pops their mother from the other end! Note that this function, like the square function above, always delivers a *single* output for each input (assuming that each person has exactly one mother). Also, a given input always produces the *same* output.

Names within names can generally be understood as representing functions. The inner name specifies the function's input, and the outer name refers to the output. Some common functions in English are:

x's father
x's favourite city (or whatever)
x's nose

The following cannot be understood as functions, since there is often either more or less than one ‘output’ object:

x’s ear
x’s friend
x’s dog
x’s car

Richard Johns
Updated Jan 2016