PHIL 114: Introduction to Symbolic Logic

From Blocks-Worlds to Extensional Interpretations

The goal is to be able to generate, for any blocks-world you can supply to establish some semantic/logical fact about a collection of blocks-language sentences, an extensional, non-blocks-world interpretation that does the same work (gets the same result as the blocks-world) but does not rely on treating the predicates in the sentences you are working with as having any particular meanings (i.e., as being about blocks arranged on a board—or about anything else). You want each such extensional interpretation to instead have a set of numbers as its domain, D.

1. To figure out how big to make the domain of numbers you want to use instead of the collection of blocks in some world, map each block in the world to a separate counting number (starting with 1) by going through the rows of the chess board, square by square, from left to right, starting with the back/upper left corner of the board (jumping all the way to the left again when you finish one row and move to the row below it). When you have gone through the whole board, put all the resulting counting numbers in your domain, D. So D is a set of individual objects (here, individual numbers).

2. Next, if a block in your blocks-world has a name (or multiple names) there, assign the number to which you mapped that block as the new object that the name now denotes in your extensional interpretation (as in: ‘a’ denotes 1, ‘b’ denotes 2, etc.). Note: more than one name can denote the same number, just as more than one name can pick out a single block. So, you might continue the above assignment of names by assigning ‘c’ as also denoting 1, if the first block in your world has both ‘a’ and ‘c’ as names. (You must also assign multiple names to a single number whenever your interpretation has to make an identity claim, e.g., ‘a = c’, true.)

3. Next, using your new domain of numbers as the objects that the predicates used in the sentence get applied to, you will assign to each one-place predicate some set of those objects (including, possibly, the empty set) as the predicate’s extension. Two-place predicates get assigned extensions that are sets (including, possibly, the empty set) of ordered-pairs of objects from D; three-place predicates get assigned extensions that are sets (including, possibly, the empty set) of ordered-triples of objects from D (and so on, with n-place predicates having sets of ordered n-tuples as their extensions). Still using the blocks-predicates, but now just as syntactic items without meanings, assign to each of them a set-of-(ordered n-tuples of)-numbers as its extension, in a way that correlates with what blocks (or ordered pairs of blocks, or ordered triples of blocks) in the world satisfy the blocks-predicates (and the identity-predicate) employed in the sentences with which you are working.

4. For example, imagine you are looking at a blocks-world that you made that has a large cube labeled as ‘b’ in the back left-corner square and a small tetrahedron labeled as ‘a’ in the fifth square of the fourth row. Imagine further that you made this world to make every member of the following set of sentences true: { (Cube(b) ∧ FrontOf(a, b)), (∃x Dodec(x) → ∃y Tet(y)), (Larger(b, a) ↔ Large(b)) }. You first map each block on the board to a counting number. The large cube gets mapped to 1; the small tetrahedron gets mapped to 2. That’s all the blocks on the board, so D = {1, 2}. Next, re-assign the names. So, ‘a’ now denotes the number 2, and ‘b’ now denotes the number 1. Now assign to each
predicate a set-of-(ordered n-tuples of)-numbers as its extension, in place of having the predicates describe blocks and relations between blocks. So, ‘Cube’ will have the number 1 in its extension, and that’s it. Same holds for the predicate ‘Large’. The predicate ‘Tet’ will have the number 2 in its extension, and that’s it. The predicate ‘Dodec’ has nothing in its extension (since there is no dodecahedron in this blocks-world) — its extension is the empty set. Note that this makes ‘∃xDodec(x)’ false, which, in turn makes ‘(∃xDodec(x) → ∃yTet(y))’ true. The predicate ‘FrontOf’ will have just the ordered-pair <2, 1> in its extension, whereas the predicate ‘Larger’ will have the ordered-pair <1, 2> as the only thing in its extension. Notice that, although I described the blocks-world as having a small tetrahedron in it, you don’t assign any extension to the predicate ‘Small’ in the extensional interpretation because this predicate is not used in any of the sentences in the set. Writing out this interpretation, I, in standard form, yields the following.

I:  
D = {1, 2}  
‘a’ denotes 2  
‘b’ denotes 1  
Cube: {1}  
Dodec: { } (or you can write ‘{ }’)  
Tet: {2}  
Large: {1}  
FrontOf: {<2,1>}  
Larger: {<1,2>}

5. This extensional interpretation is one that shows that the set of sentences can all be true together (they are consistent) as a matter of formal logic—that is, independently of the fact that it employs blocks-language predicates with the antecedent meanings they get. In fact, if the blocks-predicates were systematically replaced with meaningless predicates (of the appropriate “arities”, e.g., ‘F(_)’ for ‘Cube(_’)’, ‘J(_,_)’ for ‘FrontOf(_,_)’, etc.), a related extensional interpretation, I’, would show the resulting set of non-blocks-language sentences to be consistent. So, if the set of sentences were, e.g., { (F(b) ∧ J(a, b)), (∃xG(x) → ∃yH(y)), (K(b, a) ↔ E(b)) }, the following interpretation would show it is a consistent set.

I’:  
D = {1, 2}  
‘a’ denotes 2  
‘b’ denotes 1  
F: {1}  
G: { } (or { })  
H: {2}  
E: {1}  
J: {<2,1>}  
K: {<1,2>}

6. Constructing extensional interpretations is a way of proving what I call the “finite” logical facts: the invalidity of an argument, i.e., the absence of logical consequence between some set of sentences, Γ, and a further sentence, φ; the logical inequivalence of two sentences, φ and ψ; the consistency of a set of sentences, the contingency of a single sentence (by constructing two different extensional interpretations: one making the sentence true and one making the sentence false). Eventually, you want to be able to construct extensional interpretations directly (without starting with a blocks-world), but
the first step is to see that any blocks-worlds you have built to prove some logical fact can be mapped over to an extensional interpretation.