Limitations of the Blocks Language

Describing Size and Shape

The blocks language seems like a good one for describing a blocks world. After all, the objects in a blocks world come in just three shapes and just three sizes, and we have a predicate for each of those shapes and sizes. The arity 1 predicates are thus fully up to the task of completely and fully describing the size and shape of any block. For example, if *b* is a medium dodecahedron, the blocks language sentence Dodec(b) ∧ Medium(b) completely describes its shape and size. There is no aspect of *b*'s shape or size that is not captured by that sentence.

The shortcomings of the blocks language emerge, however, when we consider the predicates of arites 2 and 3. Apart from Larger, Smaller, and =, these all describe the locations of the blocks. But unlike the arity 1 predicates, they do not describe these locations precisely.

Describing Location

The lack of precision of the blocks language location predicates stems from that they locate the blocks only relative to one another, but not absolutely. So, for example, a sentence like LeftOf(b, c) does not tell us exactly where *b* and *c* are located, but only that *b* is somewhere to the left of *c*. By conjoining other location predicates we can narrow down the options somewhat. For example, the sentence LeftOf(b, c) ∧ FrontOf(b, c) further limits the number of different locations on the chess board where *b* and *c* might be found. But it still leaves their precise location unspecified. Perhaps *b* is in the front row and *c* in the second, or perhaps *b* is in the third row and *c* in the fifth, etc.

Precise Locations

In English, of course, we can state the precise locations of blocks. Instead of saying merely that *b* is to the left of and in front of *c*, we can say exactly where *b* and *c* are located. For example, we can say that *b* is in the first column and the second row, while *c* is in the fourth column and the third row. Or, equivalently, by taking advantage of the chess conventions of naming squares on the board, we can say that *b* is at a2 and *c* is at d3. This give us precise information about the locations of blocks *b* and *c*.

Why the blocks language cannot do this

This limitation of the blocks language has to do with what we can name in the language and what we allow to be the values of its variables. In the blocks language, we name and "quantify over" blocks, but not squares. In English, by contrast, we can name squares (as in "Square b6 is vacant") and quantify over them (as in "There is an empty square immediately to the left of cube *d*"). But in the blocks language, the names are all reserved for blocks, and a quantifier such as ∃x always means "There is a block, *x* …"

Consequences of this limitation

Given any blocks world, can we give a complete and definitive description of it? That is, can we produce a sentence (or set of sentences) which comes out true in that world but in no other? If English is our language, we certainly can. We can simply go through the sixty-four squares on the board, from a1 to h8, and state the size and shape of the block, if any, occupying it. For example, open Ackermann's World (in the TW Exercise Files folder). You will note that the following description fits Ackermann's World and no other: "There is a small tetrahedron named *a* at b2, a medium dodecahedron named *c* at c7, a small cube named *b* at e3, and a large tetrahedron named *d* at g4; there are no other blocks." But there is no way that we can produce a set of blocks language sentences that do the same thing. We can certainly specify a range of worlds containing these four blocks with the appropriate names, shapes, and sizes, but we cannot specify their precise location. We can say that *a* is to the left and in front of all the others, that *c* is in back of all the others, that *d* is to the right of all the others, that no block is between any two blocks, etc., but that would only establish, at best, the locations of the blocks relative to one another. Any set of sentences that is satisfied in Ackermann's World would also come out true in other worlds; for example, in one that results from Ackermann's World by moving each block exactly one square to the right. The spatial relationships are preserved, but the world is different; for in the new world, there is a tetrahedron in column h, whereas in Ackermann's World that column is empty.

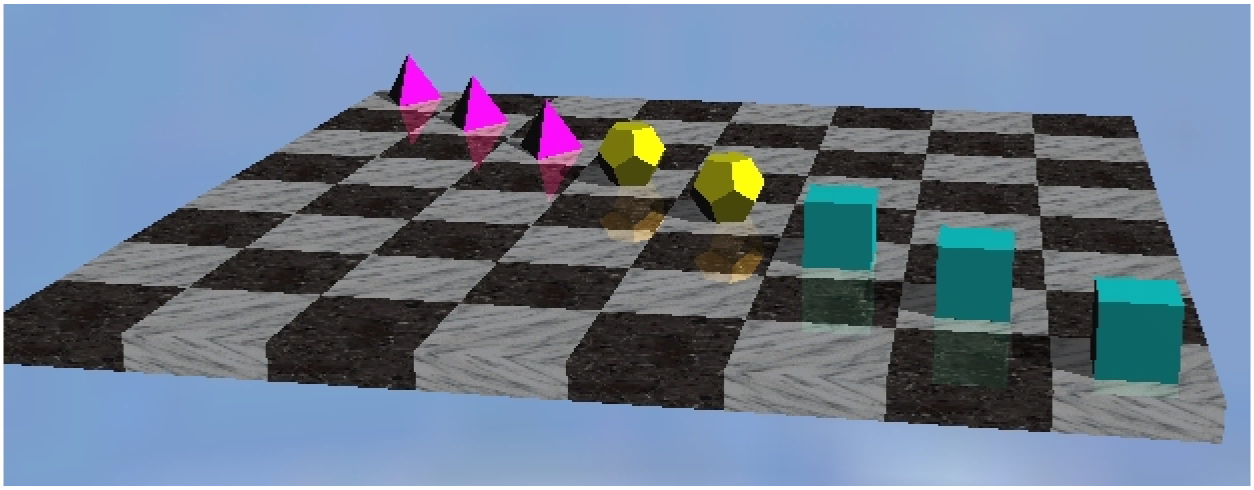
The point is even more obvious if you consider a world containing a single large cube. What could be said about such a world in the blocks language? Since there are no other blocks in the world, there is nothing in relation to which to specify the location of the large cube. It could be anywhere. The most complete description of this world that we can provide in the blocks language is:

$$\exists x\ \forall y\ (y = x \wedge \text{Cube}(x) \wedge \text{Large}(x))$$

which says that there is exactly one block, and it is a large cube. But there are sixty-four different worlds which satisfy this sentence.

Worlds completely specifiable in the blocks language

Although most worlds cannot be completely described in the blocks language, there are some that can be. They are worlds that have enough distinctive features so that when we specify the relative location of the blocks using the blocks language, we find that there is a unique absolute location for the blocks that will make all of our blocks language sentences true. Here is one such world:



Notice that this world contains exactly three cubes, exactly two dodecahedra, and exactly three tetrahedra. Every block is small. Every cube is in front of every dodecahedron. Every dodecahedron is in front of every tetrahedron. Finally, one block is in back of another just in case it is to the left of it. All of these observations about this world can be expressed in FOL sentences of the blocks language.

To confirm that this is so, open the file UniqueWorld.wld (on the Supplementary Exercises web page) and begin a new sentence file. Write FOL translations for the seven observations made above, and check to see that all of your sentences come out true. Now try making any change at all to this world, by either changing the size or shape of a block, moving a block, or adding or deleting a block.

Notice that there is no way to make any change to this world without falsifying one of the sentences. (For the purposes of this exercise, we will not count naming one of the unnamed blocks as making a change in the world.) If you do not get this result, one of your translations is wrong, and you'll have to go back and fix it. If necessary, consult the file UniqueWorld.sen to see what the correct translations are. The reason why there is no way to move the blocks while maintaining their relative locations—they cannot be slid as a group either left, or right, or forward, or back—is that every column and every row is occupied by some block or other. This last observation—that every column and every row is occupied—is one that we cannot express in the blocks language, but we have found a way to write a set of FOL sentences that requires any world in which they are all true to have no vacant columns or rows.

Finally, notice that just because our sentences here determine exactly one world which satisfies them all, we have not shown that these are the only sentences that determine that world. Trivially, of course, we could simply add an irrelevant true sentence to our set—e.g., $\exists x\ \exists y\ \mathrm{LeftOf}(x, y)$, or $\forall x\ (\mathrm{Large}(x) \rightarrow \mathrm{Cube}(x))$. But more interestingly, we could replace one of our sentences with a non-equivalent one and still have the resulting set of sentences determine the same world. You might want to look at your sentences again and see whether you can figure out how to do this. If so, stop reading right now. If not, try this: replace the fourth and fifth sentences with these: *Every cube is to the right of every dodecahedron*, and *every dodecahedron is to the right of every tetrahedron*. It's a different way of expressing things, but it still results in an identical diagonal line of blocks.

Just for fun, try writing your own set of sentences that is uniquely satisfied by a single world, and then build the world specified by those sentences. Check to make sure that any change in the world falsifies at least one of the sentences. This is a good exercise in learning how to use FOL to express conditions on worlds.