Chapter 1: Atomic Sentences

We begin by describing a simple artificial language. It is technically what is known as a first-order language; we'll call it FOL.

Although FOL can contain a much larger vocabulary than we'll use, we'll mostly be restricting ourselves to a portion of FOL that can be used to describe the worlds we can build using the software program **Tarski's World**.

At the very simplest level, FOL contains sentences that contain two kinds of ingredients: individual constants (names) and predicate symbols (property and relation words). Examples will make this clearer.

§1.1 Individual constants

a, b, c, d, e, f, n_1, n_2, n_3, … etc.

We will use these as the names of the various blocks that inhabit the "Tarski worlds" we will be examining. If we use one of these constants in describing a Tarski world, it must name some actually existing block — a block that exists in the world that we are evaluating. Note these requirements:

- Every world must contain at least one block.
- Any name that we use must name some block.
- In a given Tarski world, no name refers to more than one block.
- A block may have more than one name.
- Some blocks may not have names.

§1.2 Predicate symbols

They are listed on p. 21. Notice that they come in three "arities."

Arity 1:

Cube, Tet, Large, …etc.

Correspond to property words *is a cube*, *is a tetrahedron, is large*, etc.

Arity 2:

Smaller, Larger, LeftOf, SameSize, etc.

Correspond to relational words *is smaller than*, *is larger than*, *is to the left of*, *is the same size as*, etc.

Arity 3:

Between

Correspond to relational words *is between … and …*.

Notice that the "arity" of a predicate is the same as the **number** of individual constants (names) it takes to combine with the predicate to form a complete sentence.

§1.3 Atomic sentences

We write atomic sentences in the blocks language by combining a predicate (which always begins with a capital letter), followed by (in parentheses) one or more individual constants (which always begin with a lower case letter). The **number** of individual constants matches the **arity** of the predicate. For examples, look at the chart on p. 22.

Note that in writing atomic sentences, we use "prefix" notation:

Cube(a) Larger(b, c)

the predicate comes first, followed by (in parentheses) one or more names. The one exception is in the case of the identity symbol, =. In this case, we use "infix" notation: a = b, rather than =(a, b).

Now it is time to start learning about atomic sentences, and when they are true and when they are false. It is also time to start learning about the program *Tarski's World.*

Introduction to Tarski's World

1. Open the program. Click *Start*, *Programs*, *LPL Software*, *Tarski's World 5.6*. (Alternatively, click on *Tarski.exe*. It's in the Tarski's World Folder, inside the LPL Software Folder.) You will find an empty world and an empty sentence file. In the world, add two blocks, of different shapes and sizes. E.g., a **large cube** and a **small tetrahedron**.
2. Name them *a* and *b*. (To name the cube *a*, select the cube, then put a check-mark in the box labeled a in the Inspector and click on OK.)
3. In a new sentence file, write four sentences, describing their size and shape. You may type them in, or click on the keyboard on the screen (faster, easier, and more accurate than typing), or even copy (**`ctrl-C`**)-and-paste (**`ctrl-V`**) from another program.

Cube(a)	Tet(b)
Large(a)	Small(b)

4. Now, verify the four sentences (**`ctrl-F`**). The letter T that appears next to each sentence tells you that the sentence makes a true statement about the world you have constructed.
5. Return to the world. Alter the shapes and sizes of the blocks, as to make all of the sentences false.
6. Back to the sentences. Change them so that they still describe the shapes and sizes of the blocks, but make them all true.
7. Notice the shape and size of *a*. Add another block of that size and shape, and name it *c*.
8. Now write the sentence a = c.
9. Predict its truth-value: do you expect it to be true? If so, why? Do you expect it to be false? If so, why?
10. Verify the sentences again. Now you know that a = c is false. What can you do to the world to make it true?
11. Now you see that for a = c to be true, a and c have to name the same block. The equal sign (=) means "one and the same object," not "two objects that are exactly alike."

12. Write these additional sentences:

    Adjoins(a, b) FrontOf(a, b)

    SameSize(a, b) Between(a, b, d)

13. Now play with the blocks (move them around) and verify the sentences (**`ctrl-F`**) each time you move them. That way, you'll see under what conditions these sentences are true, and learn first-hand the meanings of the predicates. Write some more sentences, using the other predicates in the blocks language, and continue to experiment.

14. You will notice, for example:

    - Adjoins(a, b) requires that *a* and *b* be on squares that share a side; they cannot be "diagonally" adjacent.
    - FrontOf(a, b) requires no more than that *a* be closer to the front than *b*; it does not have to be anywhere near *b*, or even in the same row or column.
    - A sentence containing a name that does not name any block in a given world does not have any truth value in that world. To make Between(a, b, d) have a truth value, we had to assign the name d to one of the blocks in the world.
    - Between(a, b, d) requires that *a*, *b*, and *d* be in a straight line: either in the same row, column, or diagonal. Note that it is the **first** named block (*a* in the sentence above) that is the one in the middle.
    - If you try to move blocks in such a way that a large block adjoins another block, you cannot do it! In *Tarski's World*, no large block can adjoin any other block. (That is because the large blocks are so large they overlap their borders and infringe on the adjacent block.)
    - SameSize(a, a), SameShape(a, a), SameRow(a, a), SameCol(a, a), and a = a are always true.
    - Larger(a, a), Smaller(a, a), Adjoins(a, a), FrontOf(a, a), BackOf(a, a), RightOf(a, a), LeftOf(a, a), and a ≠ a are always false.
    - Between(a, a, a), Between(a, a, b), Between(a, b, a), and Between(b, a, a), etc. are always false. A Between sentence cannot be true unless it contains **three different names**. (Although even then it may still be false.)

15. These facts all express features of the **meanings** of the predicates in the blocks language, which closely (although not exactly) match the meanings of their English counterparts. For example, it is part of the meaning of *larger than* that a thing cannot be larger than itself; it is part of the meaning of *is in the same row as* that a thing cannot fail to be in the same row as itself.

16. The predicates of the blocks language are **determinate**, not vague. There is no gradation of sizes between small and medium, and any two objects that are both are small are considered to be of the same size. Hence, every sentence of the blocks language is either true or false. Nor are there degrees of truth and falsity—a sentence is either (entirely) true or (entirely) false, and no true sentence is "truer" than another.

Be sure to do the **You try it** on p. 24.

§1.4 General first-order languages

The blocks language is just one among many possible first-order languages. It is easy to design such a language, and there is a lot of flexibility in their design. See p. 28 for a good example of this. If you want a version of FOL in which you can say something that means, roughly, *Claire gave Scruffy to Max*, you might choose either of these:

Gave(claire, scruffy, max)

GaveScruffy(claire, max)

The first is more complicated (it has a predicate of arity 3) but gives you more flexibility—you can use it to say *Claire gave Carl to Max* simply by adding the name Carl. The second is simpler (its predicate is of arity 2) but less flexible. To write the statement about Carl using this language we'd need a new predicate: GaveCarl.

Which is the appropriate version of FOL depends on the context. The first version (arity 3) is more powerful, but that power is not always necessary. To exhibit the logical relationship between these two sentences:

Claire gave Scruffy to Max.
Claire gave Scruffy to someone.

the less complicated translation into FOL is adequate:

GaveScruffy(claire, max).

Indeed, we could even use a unary predicate (a predicate of "arity" 1), although it would be a more complicated predicate: ClaireGaveScruffyTo(max).

But neither of these translations into FOL will exhibit the logical relationship between these two sentences:

Claire gave Scruffy to Max
Claire gave something to someone.

In this case we cannot use the predicate GaveScruffy with names claire and max. We need to introduce Scruffy as a name, and use one of these translations:

Gave(claire, scruffy, max)

ClaireGave(scruffy, max).

This is because of the following important fact: FOL does not assume any connection in meaning among the predicates ClaireGave, GaveScruffy, and Gave.

Finally, to exhibit the logical relationship between these two sentences:

Claire gave Scruffy to Max.
Someone gave something to someone.

we must use a 3-place predicate in our translation:

Gave(claire, scruffy, max).

No one-place or two-place predicate will do. We will see exactly why this is so later in the course, when we deal with the logic of such expressions as *someone* and *something*.