3.12. Formal Semantics: Sentence Letters

The fundamental principle of the semantic theory is the Principle of Bivalence.

> **Principle of Bivalence**: in any possible situation, a sentence will either be *True* in that situation, or else it will be *False* in that situation (not *both*).

So it's possible for the sentence "Zebras exists" to be true (indeed, it's actually so), and it's possible for that sentence to be false (in a zebra-less situation). What isn't possible, according to Bivalence, is for the sentence to be *both true and false* in the same situation; nor to be *neither true nor false*.

The semantics of sentence letters follows directly from Bivalence, since this principle says of sentence letter "P" what it says of any sentence: there are two kinds of possibilities for that sentence.

Situations where "P" is true
Situations where "P" is false

The same point is put more compactly by depicting the first situation with a "1," and the second with "0".

P
1
0

(Keep in mind: as used here, the symbols "1" and "0" are not names of numbers – just abstract depictions of different possibilities concerning "P".)

Each such 'kind of possible situation' is called a **valuation**. So it takes two different valuations to cover all the possibilities for sentence letter "P".

The same point holds for *any* sentence letter. Two valuations exhaust the possibilities for "Q" as well.

Q
1
0

Since each sentence letter can be true or false, independently of the other, "P" and "Q" taken together yield **four** different possibilities.

P	**Q**
1	1
1	0
0	1
0	0

It takes **eight** different valuations to cover all the possibilities when taking the three sentence letters "P," "Q," and "R" together.

P	**Q**	**R**
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

We note a pattern here.

# of sentence letters	# of valuations needed to cover all the possibilities:
1	2
2	4
3	8

In general, for some number, **n**, of sentence letters, we will need $\mathbf{2^n}$ different valuations to exhaust the semantic possibilities for those sentences: 4 sentence letters require 16 valuations, 5 require 32 valuations, and so on.

(Why 2? Because of Bivalence: each sentence letter has two possible settings, true and false; and each sentence letter is true or false independent of the others.)

An easy way to calculate the number of valuations needed to cover all possibilities is to put a “2” above each sentence letter, and multiply these 2’s together – as in the earlier result for two sentence letters.

2 x 2 = 4

P	**Q**
1	1
1	0
0	1
0	0

Likewise for three sentence letters.

2 x 2 x 2 = 8

P	**Q**	**R**
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

‘Covering all the possibilities’ is important for a test of validity, since failure to find a validity counterexample is only informative when we’re sure we’ve considered every possibility, leaving no stone unturned. It was doubts about whether we’d covered every possibility that limited the effectiveness of our informal, imagination-based search for counterexamples. The Principle of Bivalence, along with some simple combinatoric patterns, will leave the formal semantics immune from such doubts.

The sentence letters are listed in alphabetic order. Starting with the right-most sentence letter (here, "R"), "1" and then "0" are written repeatedly for the required number of valuations (here, 8).

2 x 2 x 2 = 8

P	Q	R
		1
		0
		1
		0
		1
		0
		1
		0

On finishing with each sentence letter, a simple rule applies: **if there is another letter to its left, double the number** of 1's and 0's for that next letter.

There *is* a letter to the right of "R", namely "Q". While "R" alternated a single "1" and "0" repeatedly, "Q" alternates **two** 1's, and two 0's.

P	Q	R
	1	1
	1	0
	0	1
	0	0
	1	1
	1	0
	0	1
	0	0

The same rule applies when finished with “Q”: if there is a further sentence letter to the left, *double up* the 1’s and 0’s. So where “Q” alternated a *pair* of 1’s with a pair of 0’s, the next letter, “P,” uses groups of *four* 1’s and *four* 0’s – for the required number of valuations.[1]

P	Q	R
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

Each vertical table of 1’s and 0’s is a little profile of the truth-and-falsehood behavior of a sentence, called a **truth table**.

Our goal in semantics is the ability to build a semantic profile – a truth table – for *any* sentence of any type, large or small. While the three types of molecular sentences remain, we have finished the semantics for sentence letters. We noted before that a sentence letter, as a linguistic atom, is the simplest formal sentence in terms of construction. Now we see that a sentence letter is equally simple semantically: it’s either true, or it’s false, and that’s it.

[1] Fans of binary notation will note that, read from the top, the array of 1’s and 0’s counts down from 7 (“111”) to 0 (“000”). This plays no role in our semantics; it’s just an amusing side-effect.