2.22. Old Rules, New Notation

1. New Notation. Having seen the indirect semantic test offer a savings in labor compared to traditional truth tables, we propose upgrading as well the notation used to depict different possible situations, or valuations. Our goal is twofold: a further savings in labor when compared to truth table notation, but also a notation that best suits the way the indirect test approaches arguments.

To understand that second comment, consider that truth tables and the indirect test grab an argument from opposite ends. A truth table starts with values for the smallest parts of the argument – the sentence letters – and ends with values for the completed premises and conclusion. By contrast, the indirect test starts by picturing a validity counterexample for the whole argument, and follows that assumption down through the smaller parts of these sentences – ultimately, to the values of the sentence letters. Semantically, truth tables move from parts to wholes, while the indirect test moves from wholes to parts.

A simple change in how we depict possible situations serves both these ends. In truth tables we formally depict a possible situation by writing each sentence (at the top of the truth table), and putting a 1 or 0 beneath each sentence. So a valuation where "P" is true and "Q" is false is portrayed like this.

P	**Q**
1	0

We now trade in that 1/0 notation for a **vertical line**, with the following understanding: any sentence on the **left of that line** is **true**, while any sentence to the **right of the line** is **false**. In this vertical line notation, the situation where "P" is true and "Q" false looks like this.

P	
	Q

The savings this notation affords us are obvious: while depicting that situation in 1/0 notation requires 4 steps – writing the two sentences, then a 1 or 0 for each – with the new notation we need only write the sentences, since which side a sentence is on tells us whether it’s true or false. Even if we count the vertical line as a separate step, the new notation sets out the valuation in 3 steps, compared to the truth table’s 4.

Those savings compound as we heap on more sentences. For instance, to build a valuation for 10 sentence letters using a truth table calls for 20 steps (each sentence, and a 1 or 0 beneath it), while the new notation needs only 11 steps: the 10 sentences, and the vertical line. The savings in labor approaches half.

2. Indirect Test Meets Vertical Line: Semantic Rules Revisited (and Reversed). The indirect test of validity begins by picturing a situation where all the premises are true, but the conclusion is false. In vertical line notation that would look like this.

Premises	
	Conclusion

Suppose, for instance, we test this familiar argument for validity.

$$
\begin{array}{l}
(P \vee Q) \\
\sim P \\
\hline
\therefore\ Q
\end{array}
$$

The indirect test begins like so.

$(P \vee Q)$	
$\sim P$	
	Q

But here our test hits a new snag. For we next follow the consequences of these values down through the parts of the sentences, guided by the semantic rules. The problem: **the semantic rules are stated in the bad old 1/0 notation**.

We'll get no further with this test until we restate those rules in vertical line format – and, once again, in a way that moves (like the indirect test) from wholes to parts. Make no mistake: we don't wish to change **what the semantic rules say** (since we suppose these rules get it right concerning when sentences are true, and when false) – just **how** the rules state those semantic facts.

We begin with the Negation Rule.

▲	~▲
1	0
0	1

Moving from whole ("~▲") to part ("▲") in this semantic rule is simple: we just read each valuation **from right to left**.

Read this way, the first valuation says: if “~▲” is false, then “▲” is true.

	▲	~▲	
⇨	**1**	**0**	⇦
	0	1	

We say this with a vertical line like so.

False Negation

	~▲ ✓
▲	

And each time we chase a semantic consequence from whole to part, we **check** that whole. For when testing for validity, it will be essential to chase down **every** consequence – in the same way that, in truth tables, the test of validity relied on going through **every** valuation. Checking each sentence in this way helps us keep track of which sentences we’ve examined.

The other half of the Negation Rule says: if “~▲” is true, “▲” is false.

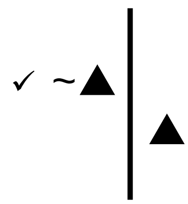
	▲	~▲	
	1	**0**	
⇨	0	1	⇦

On a vertical line that reads like so.

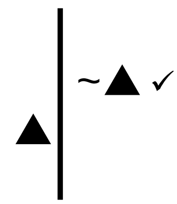
✓ ~▲	
	▲

We now have our restatement of the semantic Negation Rule, in the new format.

True Negation



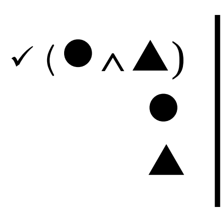
False Negation



Turning to the Conjunction Rule, true conjunctions are equally simple: **if the whole conjunction is true, both parts are true**.

	●	▲	(● ∧ ▲)
⇨	**1**	**1**	**1**
	1	0	0
	0	1	0
	0	0	0

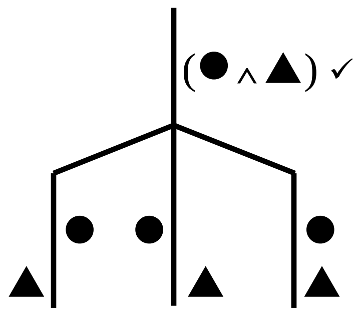
In new notation that looks like so.



A false conjunction is more puzzling. There are three *different* ways a conjunction could be false: by having the left part false, having the right part false, or having both parts false.

	●	▲	(● ∧ ▲)	
	1	1	1	
⇨	1	0	**0**	⇦
⇨	0	1	**0**	⇦
⇨	0	0	**0**	⇦

We *could* restate this in vertical lines by splitting the original line into three, to cover the different possibilities.



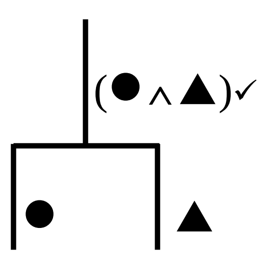
But we can state this information more economically by eliminating the overlaps and repetition in the truth table. In fact there are only **two** ways a conjunction would be false: either because its left part is false, or because its right part is. (The last valuation, where both parts are false, is just the overlap of those two options.)

●	▲	(● ∧ ▲)
1	1	1
1	**0**	**0** ⇦
0	1	**0** ⇦
0	0	**0** ⇦

As a general rule: **whenever a conjunction is false, either its left or right part is false (possibly both)**.

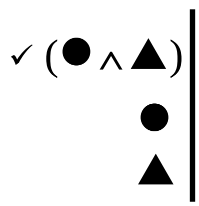
In new notation we split the original tree path into two, in order to represent these two possibilities.

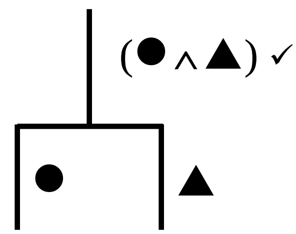
False Conjunction



Retooled in this way, the semantic Conjunction Rule reads as follows.

True Conjunction **False Conjunction**

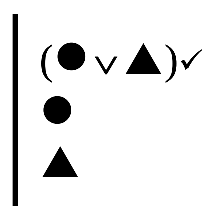




With disjunctions, the false case is simplest: **a disjunction is only false when *both* its parts are false**.

●	▲	(● ∨ ▲)
1	1	1
1	0	1
0	1	1
⇨ **0**	**0**	**0** ⇦

That’s easy to state with a vertical line.



By contrast, there are *three* ways of making a disjunction **true**.

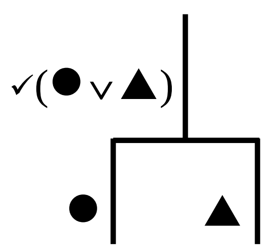
●	▲	(● ∨ ▲)
1	1	**1** ⇦
1	0	**1** ⇦
0	1	**1** ⇦
0	0	0

But as with the Conjunction Rule, we can distill these three into two essential cases: **whenever a disjunction is true, either its left part is true, or its right part is** – possibly both. (The first valuation is the overlap of these two.)

●	▲	(● ∨ ▲)
1	**1**	**1** ⇦
0	**1**	**1** ⇦
1	**0**	**1** ⇦
0	0	0

With a true disjunction, the tree path branches to cover the two different possibilities.

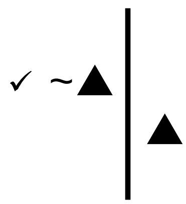
True Disjunction



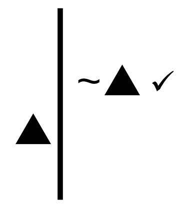
With the semantic rules translated into vertical line format, we are at last in a position to enjoy a maximally efficient test of validity: the indirect test, executed in improved notation.

Semantic Rules (Reformatted)

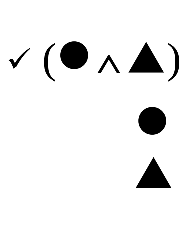
True Negation



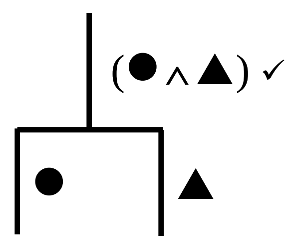
False Negation



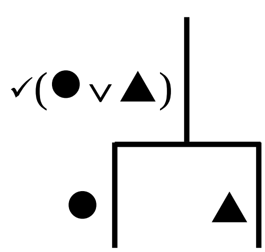
True Conjunction



False Conjunction



True Disjunction



False Disjunction