2.30. Expressive Adequacy: Further Languages

Earlier considerations of expressive adequacy[1] focused on the formal language of Chapter Two – the language of {~, ∧, ∨}, plus sentence letters.[2] In what follows we look at various '**sub-languages**': formal languages got by casting out one or more of the connectives of the Chapter Two language.

Beginning with the Chapter Two set of connectives {~, ∧, ∨}, removal of one or more connectives yields the following six (non-empty) subsets.

{~, ∧}	{~}
{~, ∨}	{∧}
{∧, ∨}	{∨}

Each of these (along with sentence letters) constitutes a formal language. And here the question of expressive adequacy arises again: we ask, for each of these six languages, whether **every possible** truth table is matched by some sentence in that language. If so, the language is expressively adequate – capable of pairing each truth table with some sentence of that language, just as the Chapter Two language does. If not, there is some truth table which that formal language has no matching sentence for.

It turns out that some of these formal languages are expressively adequate, but that others are provably inadequate. In what follows we establish which result holds for each of these six formal languages.

Our prior experience establishing the expressive adequacy of {~, ∧, ∨} already gives us a good idea which sort of approach will work in this case – and which won't. Specifically: for establishing that a language is expressively adequate, it's no good trying to list all possible truth tables, one by one, and finding for each a matching sentence. Since there are an **infinite** number of truth tables, such a

[1] In 2.28.

[2] The language {~, ∧, ∨} also features parentheses; but as mere punctuation these go without mention in the discussion that follows.

piecemeal matching process will never end. Likewise, in attempting to show that a language is expressively inadequate, it’s pointless to single out some truth table and try to show that each sentence in the language fails to match it. For again, since there are an infinite number of sentences in each of the formal languages we’re considering, comparison of sentences to that truth table will go on forever.

To prove the expressive **adequacy** of {~, ∧, ∨} we built a **general procedure** for constructing a {~, ∧, ∨} sentence, for any given truth table. And proving the expressive adequacy of further formal languages will likewise rely on a (modified form of) this general procedure.

As we’ll see, establishing the expressive **inadequacy** of a formal language calls instead for finding a **distinctive semantic feature** of all truth tables generated by that language, and showing that some truth table lacks this feature.

1. The Languages {~, ∧} and {~, ∨}. The language {~, ∧} is identical to the formal language of Chapter Two except for lacking vels. Now for any sentence letter, or larger sentence featuring just sentence letters, tildes and/or wedges, {~, ∧} can of course build that sentence just as well as {~, ∧, ∨}. For all those sentences, {~, ∧} will construct the sentence the same way that {~, ∧, ∨} did; so the truth table for that sentence will be step-for-step identical as well.

If {~, ∧} loses any expressive power – if there is indeed some truth table for which {~, ∧} can offer no matching sentence – it could only be owing to its lack of a vel.

The semantic contribution made by the vel is summed up in the semantic Disjunction Rule: combining two smaller sentences with a vel yields a sentence **true as long as at least one of these parts is true** (and so false only when both parts are false).

Disjunction Rule:

●	▲	(● ∨ ▲)
1	1	1
1	0	1
0	1	1
0	0	0

If the {~, ∧} language can provide some counterpart with this semantic behavior, then loss of the vel will be seen **not** to have impaired the semantic powers of {~, ∧}. So, starting with the two parts of the disjunction (● and ▲), we need a way of applying tildes and/or wedges that yields a sentence with the same semantic behavior as the disjunction of those parts.

Thanks to DeMorgan's Law we know that such a structure exists. For (● ∨ ▲) is logically equivalent to ~(~● ∧ ~▲).

●	▲	~●	~▲	(~● ∧ ~▲)	~(~● ∧ ~▲)
1	1	0	0	0	1
1	0	0	1	0	1
0	1	1	0	0	1
0	0	1	1	1	0

~(~● ∧ ~▲) is true as long as one of ● and ▲ are true (and so false only when both ● and ▲ are false).

That holds no matter which sentences go in the ● and ▲ spots. In the simplest case, where sentence letters such as “P” and “Q” are combined into “(P ∨ Q),” the truth table for “(P ∨ Q)” is the same as that for “~(~P ∧ ~Q)”. And the same holds with any larger molecular inputs for ● and ▲: any vel added further up the construction tree (combining larger left and right parts) will likewise be matched with the cluster of connectives “~(~__ ∧ ~__)” applied to those same parts, and guaranteed to have the same truth table as that disjunction.

So the language {~, ∧} does indeed have a structure matching the semantic contribution of the vel. That means that {~, ∧} has the same expressive power as {~, ∧, ∨}. But {~, ∧, ∨} is expressively adequate. So **{~, ∧} is expressively adequate**.

Indeed, we can provide a modified procedure for matching any truth table to a sentence in the language {~, ∧}.[3]

- If the truth table is false in every valuation, use “(P ∧ ~P)” as the matching sentence.

- If the truth table is true in exactly one valuation, build a valuation sentence true in that valuation. (*Since valuation sentences are built out of sentence letters, tildes, and wedges, they are sentences of the {~, ∧} language*.)

- If the truth table is true in more than one valuation, (i) build a valuation sentence for each **false** valuation (valuation with a ‘0’); (ii) negate each of those valuation sentences; and (iii) conjoin together all of those negated sentences.

[3] Though we don’t need to set out the steps of such a general method in order to prove {~, ∧} expressively adequate. That point is settled once we show that {~, ∧} is semantically equivalent to {~, ∧, ∨}; for we already know that {~, ∧, ∨} is expressively adequate.

As an illustration, we construct a {~, ∧} sentence to match this truth table.

?
1
0
0
1

As usual we attach truth tables for the appropriate number of sentence letters (here two letters, because there are four valuations).

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

We then construct a valuation sentence for each **false** valuation (a valuation with a 0) in the ‘mystery truth table,’ following the same procedure as before: if a letter is true in that valuation, add that letter; if the letter is false in that valuation, add the negation of that letter.

P	**Q**	**~P**	**~Q**	**(P ∧ ~Q)**	**(~P ∧ Q)**	**?**
1	1	0	0	0	0	1
1	**0**	0	1	**1**	0	**0**
0	**1**	1	0	0	**1**	**0**
0	0	1	1	0	1	1

Then we negate each valuation sentence.

P	**Q**	**~P**	**~Q**	**(P ∧ ~Q)**	**(~P ∧ Q)**	**~(P ∧ ~Q)**	**~ (~P ∧ Q)**	**?**
1	1	0	0	0	0	**1**	**1**	1
1	0	0	1	1	0	**0**	**1**	0
0	1	1	0	0	1	**1**	**0**	0
0	0	1	1	0	0	**1**	**1**	1

These negations are then conjoined together– yielding a sentence matching the mystery truth table.

P	Q	~P	~Q	(P ∧ ~Q)	(~P ∧ Q)	~(P ∧ ~Q)	~(~P ∧ Q)
1	1	0	0	0	0	1	1
1	0	0	1	1	0	0	1
0	1	1	0	0	1	1	0
0	0	1	1	0	0	1	1

(~(P ∧ ~Q) ∧ ~(~P ∧ Q))	**?**
1	1
0	0
0	0
1	1

This method will in general yield the correct result: some sentence in the {~, ∧} language, for each 'mystery truth table'.[4]

The same general strategy used to establish the adequacy of {~, ∧} will apply as well to {~, ∨}; for DeMorgan's Law also guarantees a semantic surrogate for conjunctions, using only tildes and vels as connectives. So the language {~, ∨} is also **expressively adequate**.

2. The Language {~}. The remaining languages are all expressively inadequate. That means that for each language, there is some truth table for which that language offers no matching sentence.

Of these, {~} is most obviously inadequate. And seeing why that is obvious highlights the general strategy used to establish the semantic inadequacy of a language.

[4] Since the negation of a valuation sentence is equivalent to a counter-valuation sentence, this method is a {~, ∧} variant on **Conjunctive Normal Form** (discussed in 2.29).

Only a few examples of sentence in this language, with corresponding truth tables, suffice to illustrate a general pattern for all {~} truth tables.

P	Q	~P	~Q	~~P	~~Q	~~~P	~~~Q	~~~~P
1	1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1	1
0	1	1	0	0	1	1	0	0
0	0	1	1	0	0	1	1	0

Each truth table here has the same number of 1s and 0s. And that is bound to hold in general. For (i) each sentence letter has the same number of 1s and 0s. And (ii) since the semantic rule for negations replaces each 1 with 0 and each 0 with 1, an even number of 1s and 0s into that rule yields an even number of each as output. And every sentence in this language is either a sentence letter or a negation. So every sentence in the {~} language has a truth table with the same number of 1s and 0s.

But the truth table for, e.g., "(P ∧ Q)" lacks that feature: it takes a single 1 and three 0s. And the same holds for the "(P ∨ Q)" truth table (three 1s and a single 0).

P	Q	(P ∧ Q)	(P ∨ Q)
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

These are truth tables which **no** {~} sentence will take. So the {~} language is **semantically inadequate**.

Since {~} is a subset of the Chapter Two language {~, ∧, ∨}, we see that **not every 'sub-language'** of {~, ∧, ∨} **is expressively adequate**.

3. The Languages {∧}, {∨}, and {∧, ∨}. A similar strategy shows the remaining three formal languages to be semantically inadequate.

Beginning with {∧}, the simplest truth tables for this language illustrate its semantic shortcoming.

P	**Q**	**(P ∧ P)**	**(Q ∧ Q)**	**(P ∧ Q)**	**((P ∧ Q) ∧ P)**
1	1	1	1	1	1
1	0	1	0	0	0
0	1	0	1	0	0
0	0	0	0	0	0

No matter how many sentence letters (or conjunctions of them) we conjoin together, the resulting truth table will be **true in the first valuation** (where both parts are true). Being true in the first valuation is a feature found in the simplest cases, and a feature preserved by any conjunction of parts having that feature; so it is a feature of **all** {∧} truth tables.

But some truth tables are not true in the first valuation – most obviously, the negation truth tables.

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

The truth table for “~P” is **false in the first valuation**. Since no combination of sentence letters and wedges yields a sentence false in the first valuation, this is a truth table which no {∧} sentence can match. That shows that {∧} is **expressively inadequate**.

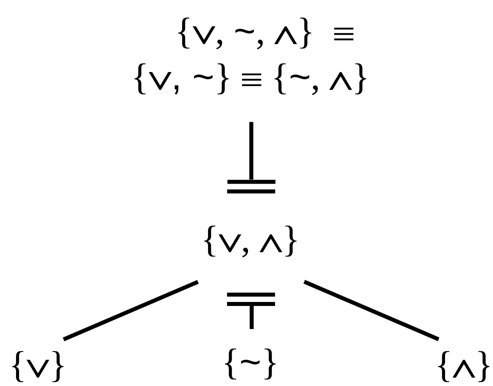
The same point holds for {∨}. For here again every truth table is true in the first valuation.

P	**Q**	**(P ∨ P)**	**(Q ∨ Q)**	**(P ∨ Q)**	**((P ∨ Q) ∨ P)**
1	1	1	1	1	1
1	0	1	0	1	1
0	1	0	1	1	1
0	0	0	0	0	0

So {∨} will not build the truth table for, say, "~P". That means {∨} is **expressively inadequate**.

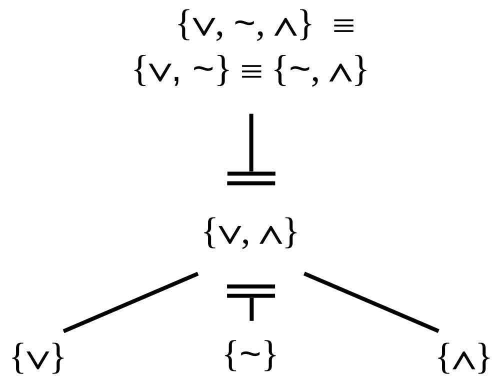
And obviously the same holds for {∧, ∨}, since any combination of sentence letters, wedges, and vels will still be true in the first valuation. So {∧, ∨} is **expressively inadequate**.

Hence we end with the formal languages arranged like so.



The three languages at the top – {∨, ~, ∧}, {∨, ~}, and {~, ∧} – are all expressively adequate, and so equivalent languages in terms of expressive power.

The languages {∨}, {~}, and {∧} are all weaker than those languages, and so expressively inadequate – though these last three language are not equivalent in expressive power since, for example the language {~} can cover a truth table that neither {∨} nor {∧} can (for instance, the truth table taken by the sentence “~P”).



And while {∨, ∧} isn’t expressively adequate – since it too offers no sentence matching the truth table for “~P” – it’s still expressively more powerful than either of its sub-languages {∨} and {∧}. For {∨} can’t built a sentence matching the truth table for “(P ∧ Q)”; and {∧} can’t build a sentence taking the truth table for “(P ∨ Q)”.