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] Here 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 {~, ∧, ∨}, removing one or more connectives yields the following six (non-empty) subsets.

{~, ∧}	{~}
{~, ∨}	{∧}
{∧, ∨}	{∨}

Each of these (along with sentence letters) constitutes a formal language. And we ask of each whether **every possible** truth table is matched by some sentence in that language. If so, the language is expressively adequate. If not, there's some truth table which that formal language has no matching sentence for, and the language is **expressively inadequate**.

It turns out some of these formal languages are expressively adequate while others are provably inadequate. Our experience establishing the expressive adequacy of {~, ∧, ∨} gives us a good idea which sort of approach will work in this case – and which won't. 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 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, there are an infinite number of sentences in each of the formal languages we're considering, so comparison of sentences to that truth table will go on forever.

[1] In 2.27.

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

To prove the expressive **adequacy** of {~, ∧, ∨} we instead 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. Establishing the expressive **inadequacy** of a formal language calls instead for finding a **distinctive semantic feature** of all sentences in 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 composed of just sentence letters, tildes and/or wedges, {~, ∧} can build that sentence just as well as {~, ∧, ∨}. 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** (hence 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 the same semantic behavior, loss of the vel will be seen not to have impaired the semantic powers of {~, ∧}.

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). And that holds no matter which sentences go in the ● and ▲ spots. So {~, ∧} 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.

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

[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.

?
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 strategy used to establish the adequacy of {~, ∧} applies to {~, ∨}; for DeMorgan's Law guarantees a semantic surrogate for conjunctions, using only tildes and vels as connectives. So the language {~, ∨} is likewise **expressively adequate**.

2. The Language {~}. The remaining languages are all expressively inadequate: for each language there's some truth table for which that language offers no matching sentence.

Of these, the case against {~} is most obvious. And seeing why 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

For each truth table here **the number of 1s is the same as the number of 0s**. And that’s bound to hold in general. For each sentence letter has the same number of 1s as of 0s; and since the semantic rule for negations replaces each 1 with 0 and each 0 with 1, the output of the rule leaves the same number of 1s as of 0s wherever the input did. So **every sentence in the {~} language has a truth table where the number of 1s is the same as the number of 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.

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 that the remaining three formal languages are semantically inadequate.

Concerning {∧}, the fundamental semantic feature of a conjunction suffices to prove the inadequacy of this language: **when** (indeed, only when) **both parts** of the conjunction **are true, the whole conjunction is true**.

Conjunction Rule

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

Now since **sentence letters are true in the first valuation**, and the semantic rule for conjunctions leaves the output (the whole conjunction) true whenever both inputs (the left and right parts of the conjunction) are, conjunctions of sentence letters will be true in the first valuation – as will conjunctions of those conjunctions, and so on.

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

Being true in the first valuation is a feature found in the simplest cases, and preserved by any conjunction of parts having that feature; so it's a feature of **all** {∧} truth tables. **Every sentence in the {∧} language is true in the first valuation**.

But some truth tables aren't true in the first valuation – for example, the negation of a sentence letter.

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

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. Hence the language {∧} is **expressively inadequate**.

The same point holds for {∨}. For here again **when both parts of the disjunction are true**, the semantic rule for disjunction makes the **whole disjunction true**.

Disjunction Rule

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

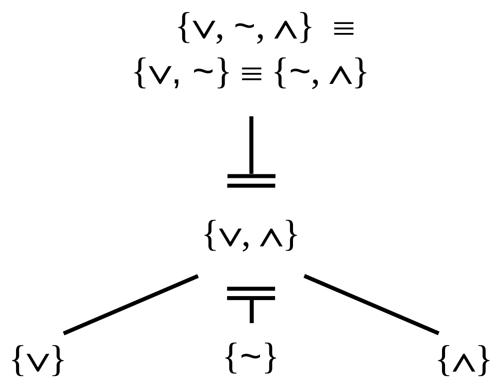
Since sentence letters are true in the first valuation, disjunctions of them (and disjunctions of those disjunctions, etc.) will be true in the first valuation.

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 the language {∨} will contains no sentence taking 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**.

Thus we end with the formal languages arranged like so.



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

The languages {∨}, {~}, and {∧} are all weaker than those languages, hence expressively inadequate – though none of the bottom three language are equivalent in expressive power. For example, the language {~} can cover a truth table that neither {∨} nor {∧} can (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 {∨} and {∧} each features sentences missing in the other language.[5]

---

[5] Note that any conjunction of sentence letters, or conjunction of those conjunctions, etc., will be **true only where every sentence letter appearing in the sentence is true**. For instance, "((P ∧ Q) ∧ (P ∧ P))" is true when, and only when, "P" and "Q" are true. For any sentence in the {∧} we could thus remove all inner parentheses and strike out duplicate letters (thanks to associativity and idempotence). But the language {∨} obviously has sentences which are true even where not every sentence letter in that sentence is true – for example, "(P ∨ Q)".

On the other hand, no disjunction of sentences letters, or disjunction of those disjunctions, etc., will be false when not all the sentence letters have the same value; but the sentence "(P ∧ Q)" is false when "P" is true and "Q" is false.