3.27. Valuation Disjunctions, Expressive Adequacy, and Disjunctive Normal Form

1. Expressive Adequacy. Because each construction rule is matched by a semantic rule, we know that each formal sentence has a truth table. Less obvious perhaps, is whether the reverse is true as well: whether each truth table – an array of 2^N 1s and/or 0s – is guaranteed to have a matching formal sentence.

If the answer is *No*, then there will be some truth table matched by no sentence in the formal language. In that case the construction rules – which provide all the sentences available in the formal language – would not keep up with the set of all possible truth tables. In the jargon of formal logic, that would make our formal language **expressively inadequate**.

But that unhappy outcome is (happily) not the case. In fact our formal language is **expressively adequate**: for any possible truth table, the formal language is guaranteed to have a matching sentence.

Proving this requires a *general* procedure which begins with a truth table, and ends with a corresponding formal sentence. (And by "corresponding" we mean: a sentence which, *according to the semantic rules*, really does take that truth table.)

2. Valuation Sentences Revisited: Valuation Disjunctions. In search of our general procedure, we return to sentences discussed in the previous section.

Note first that for an array of 1's and 0's to qualify as a truth table, it must contain $\mathbf{2^N}$ 1s and/or 0s. (A truth table can't have *three* 1s and/or 0s, or *five* of them; it must have two, or four, or eight, etc.) Our task is *only* to find a formal sentence for each genuine truth table – that is, for each array of $\mathbf{2^N}$ 1s and/or 0s. When we speak below of a "mystery truth table," we mean such an array of 1s and/or 0s.

The general sentence-matching technique begins by lining up such a ‘mystery truth table’ with truth tables for sentences letters – **N** many sentence letters, for the $\mathbf{2^N}$ valuations in the mystery truth table. So if the truth table has 4 ($2^{\mathbf{2}}$) valuations, we precede it by the truth tables for **2** sentence letters (say, “P” and “Q”); whereas if we face 8 ($2^{\mathbf{3}}$) valuations, we attach truth tables for **3** sentence letters (say, “P,” “Q,” and “R”).

So consider this mystery truth table.

?
1
0
0
1

Having 4 valuations, we attach before it truth tables for *two* sentence letters.

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

In the next step we focus on the ‘**true** valuations’ (those with a 1). For each such ‘true valuation’, we construct a sentence true in *only* that valuation.

Thanks to our earlier explorations, we know just what sort of sentence fills the bill: a **valuation sentence** is a sentence true in *exactly one* valuation. (Recall that for a given set of sentence letters, a valuation sentence is a conjunction where each sentence letter in the set appears exactly once – either as-is, or negated.)

So {P, Q} yields the following four valuation sentences.

(P ∧ Q)
(~P ∧ Q)
(P ∧ ~Q)
(~P ∧ ~Q)

As truth tables illustrate, each of these sentences is true in just one valuation.

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

And since no two of these sentences are true in the same valuation, each valuation sentence is paired with its own unique valuation. That means that any 'true valuation' – a valuation with a 1 – has a corresponding valuation sentence.

The general procedure for pairing a 'true valuation' with a valuation sentence requires us to look at values of the sentence letters in that valuation, and construct our valuation sentence accordingly.

- If the sentence letter is true in that valuation, the valuation sentence should **include that sentence letter**.

- If the sentence letter is false in that valuation, the valuation sentence should **include the negation of that sentence letter**.

So, for example, in a valuation where “P” and “Q” are both true, the corresponding valuation sentence features both these sentence letters: “(P ∧ Q)”.

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

In a valuation where “P” is true and “Q” is false, the corresponding valuation sentence will feature “P” and “~Q”.

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

Now our mystery truth table is true in the **first** and **fourth** valuations. So we build a valuation sentence to match each of these valuations: “(P ∧ Q)” for the first, and “(~P ∧ ~Q)” for the fourth.

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

The sentence matching the mystery truth table will be true in *both* the first *and* fourth valuations. Neither of our valuation sentences here fits that pattern, since each is true in only one valuation. But these two valuation sentences can figure as parts of a larger sentence which is **true whenever one of its parts is true**.

That, of course, describes the truth conditions for a **disjunction**. Sure enough: a disjunction of the two valuation sentences matches the mystery truth table.

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

Such a disjunction of valuation sentences will be called a **valuation disjunction**. And we have already shown enough to recognize the following.

> Any truth table 'true' in a single valuation has a matching **valuation sentence**. And any truth table 'true' in more than one valuation has a matching **valuation disjunction**.

A larger, eight-valuation truth table provides another illustration. The three 'true' valuations are matched with valuation sentences like so.

P	Q	R	**?**	**Valuation Sentences**
1	**1**	**1**	**1**	((P ∧ Q) ∧ R)
1	1	0	0	
1	**0**	**1**	**1**	((P ∧ ~Q) ∧ R)
1	0	0	0	
0	1	1	0	
0	**1**	**0**	**1**	((~P ∧ Q) ∧ ~R)
0	0	1	0	
0	0	0	0	

These three valuation sentences are combined into a disjunction.

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

Now while we have generally been fastidious about parentheses in the formal language, their proliferation here is an eye-boggling impediment. So for valuation sentences and disjunctions we allow this convenience: when multiple parts are disjoined together, we will delete all inner parentheses. “((P ∨ Q) ∨ R)” will then become “(P ∨ Q ∨ R)”. And the above disjunction will likewise be (mildly) simplified.

$$(((\underline{P \wedge Q) \wedge R}) \vee ((\underline{P \wedge \sim Q) \wedge R}) \vee ((\underline{\sim P \wedge Q) \wedge \sim R}))$$

We can afford this notational laxity because (as noted already in our treatment of formal translation)[1], disjunctions are **associative**: the grouping of parts in a disjunction makes no difference to truth or falsity. For instance, whenever “((P ∨ Q) ∨ R)” is true,“(P ∨ (Q ∨ R))” is true (and vice versa).

The same holds for conjunctions: whenever “((P ∧ Q) ∧ R)” is true, “(P ∧ (Q ∧ R))” is true (and vice versa). So we allow the same loosening of notation for many-part valuation sentences – permitting a further simplifying of valuation disjunctions.

$$((\underline{P \wedge Q \wedge R}) \vee (\underline{P \wedge \sim Q \wedge R}) \vee (\underline{\sim P \wedge Q \wedge \sim R}))$$

Leaving off outermost parentheses (but only when they are indeed the *outermost* of all symbols!) provides a final bit of simplification – and the final form of our valuation disjunction.

$$(\underline{P \wedge Q \wedge R}) \vee (\underline{P \wedge \sim Q \wedge R}) \vee (\underline{\sim P \wedge Q \wedge \sim R})$$

(Note that returning this sentence to official construction format is easy: because grouping does not affect truth, we can use parentheses to group the parts however we please. For example, we can always group parts in pairs, from the left, to return to the original sentence, above.)

[1] “*3.7. Constructing Formal Sentences*,” Section 2.

Truth tables confirm that this sentence does indeed take our mystery truth table.

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

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

3. Valuation Sentences, Valuation Disjunctions, and Disjunctive Normal Form. Alas, the method of valuation sentences and valuation disjunctions falls just shy of the general procedure we are looking for. If a mystery truth table is true in just one valuation, a valuation sentence is sure to match it; and if it's true in more than one, a valuation disjunction will. But that overlooks the case where a mystery truth table is **true in no valuations**. Neither valuation sentences nor valuation disjunctions are any help here, since they can never be false in *every* valuation. (In semantic jargon: every valuation sentence and valuation disjunction is bound to be *satisfiable*.)

At least two different solutions are available to close this gap – neither a large departure from the method set out above.

The **first** strategy is simply to specify a sentence to be used in this troublesome case. A truth table with 2^N 0s is a truth table for a **contradiction**; and since all contradictions are logically equivalent, they are semantically interchangeable. So we can add a rule to our method that in such a case the sentence matching the mystery truth table is, say, "(P ∧ ~P)". (Since "(P ∧ ~P)" is a sentence of the Chapter Three language, we succeed in finding a matching Chapter Three sentence for the mystery truth table.)

The general method for matching a Chapter Three sentence to each truth table will then run as follows.

- If the truth table has **no true valuations** (is false for every valuation), use “(P ∧ ~P)” as the matching sentence.

- If the truth table has **exactly one true valuation**, build a valuation sentence matching that valuation.

- If the truth table has **more than one true valuation**, build a valuation disjunction true in just those valuations.

Since every truth table is bound to fall into one of these three categories, every truth table is guaranteed a matching Chapter Three sentence.

The **second**, more traditional strategy involves relaxing our original restrictions on valuations sentences – and so, by association, on valuation disjunctions. When building a family of valuation sentence from a set of sentence letters we required that each sentence letter in the set appear *exactly once*. If that restriction is lifted, we return to the larger family of **basic conjunctions**.[2] These include all the valuation sentences of old, but also sentences such as the following.

(P ∧ ~P)
((P ∧ Q) ∧ ~P)
((P ∧ Q) ∧ ~R) ∧ ~P)

Since each of these conjunctions contains both “P” and “~P,” each yields a truth table false in every valuation; and that’s exactly the sort of case left out by valuation sentences.

We can build disjunctions of these basic conjunctions, just as we did earlier out of valuation sentences. But now some such disjunctions may have one or more parts which are contradictions. (In the limit case, where *every* part of the disjunction is a

[2] From “*3.26. Valuation and Anti-Valuation Sentences*,” Section 1.

contradiction, the entire disjunction will itself be a contradiction. For instance, "((P ∧ ~P) ∨ (Q ∧ ~Q))" is a contradiction.)

Basics, basic conjunctions, and disjunctions of them, form a family of sentences said to be in **Disjunctive Normal Form** (or "**DNF**," for short). The following construction rules offer precise conditions for being a DNF sentence.[3]

Basics:

1. Sentence letters are basics.
2. Negations of sentence letters are basics.

Basic Conjunctions:

1. Basics are basic conjunctions.
2. If ● and ▲ are basic conjunctions,
   then (● ∧ ▲) is a basic conjunction.

Sentences in Disjunctive Normal Form (DNF):

1. Basic Conjunctions are DNF sentences
2. If ● and ▲ are DNF sentences,
   then (● ∨ ▲) is a DNF sentence.

DNF sentences include contradictions such as "(P ∧ ~P)," valuation sentences, and valuations disjunctions – plus further sentences falling into none of those categories.

Yet despite the greater sentence-building power DNF offers over the earlier method of valuation disjunctions, this excess is largely irrelevant to our purposes. Since the sentences used in the first method – valuation sentences, valuation

[3] The construction rules for DNF sentences can be summed up quite simply in terms of **scope**: in DNF any vel has wider scope than a wedge, and any vel or wedge has wider scope than a tilde. (To put the same point negatively: in DNF no tilde has wider scope than a wedge or vel, and no wedge has wider scope than a vel.)

disjunctions, and contradictions such as “(P ∧ ~P)” – all qualify as DNF sentences, the procedure for finding such a DNF sentence remains unchanged.

- If the truth table is true in exactly one valuation, build a valuation sentence true in that valuation.

- If the truth table is true in more than one valuation, build a valuation disjunction true in those valuations.

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

In essence, the DNF approach over-generates wildly – allowing far more sentences than the first approach did – and then chops this jungle down to just those sentences of interest to us, through the three-part procedure above.

Since either method provides a general procedure for matching each truth table with a formal sentence, we are guaranteed that no truth table lies out of the reach of the Chapter Three language – the language of {~, ∧, ∨}, plus sentence letters. Thus the language of Chapter Three is **expressively adequate**.