2.27. Expressive Adequacy and Normal Forms

1. Expressive Adequacy. Because each construction rule is matched by a semantic rule, we know each formal sentence has a truth table. Less obvious, perhaps, is whether the reverse is also true: 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*, there will be some truth table matched by no sentence in the formal language. In that case the construction rules wouldn't keep up with the set of all possible truth tables – wouldn't, as we'll say, **cover** all those truth tables. In the jargon of formal logic, our formal language would then be **expressively inadequate**.

But that unhappy outcome is happily not the case. In fact our formal language is **expressively adequate**: for any given truth table, the formal language provides a matching sentence.

Proving that fact involves a **general** procedure which starts with a truth table and ends with a matching formal sentence. (And by "matching" we mean: a sentence which, following 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's got to have two, or four, or eight, etc.) Our task is **only** to find a formal sentence for each genuine truth table – for each array of $\mathbf{2^N}$ 1s and/or 0s. When we speak below of a "mystery truth table" we'll 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 (2^2) valuations, we precede it by truth tables for **2** sentence letters (say, “P” and “Q”); whereas if we face 8 (2^3) 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

Next we focus on the ‘**true** valuations’ (those with a 1). For each such ‘true valuation’ we construct a sentence true in **just** that valuation.

Thanks to our earlier explorations we know 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 the set {P, Q} yields the following four valuation sentences.

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

Since no two of these sentences are true in the same valuation, each valuation sentence is paired with its own unique valuation. That means any valuation with a 1 in a given column will have a corresponding valuation sentence.

The general procedure for pairing 'true valuations' and valuation sentences requires us to look at values of the sentence letters in that valuation and construct a 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**.

For example, in a valuation where "P" and "Q" are both true, the corresponding valuation sentence features both these sentence letters: "$(P \wedge Q)$".

P	**Q**	**$(P \wedge Q)$**
1	**1**	**1**
0	1	0
1	0	0
0	0	0

Whereas 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

Since our mystery truth table had 1 in the **first** and **fourth** valuations, we build a valuation sentence matching each: “(P ∧ Q)” for the first, “(~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**

A sentence matching the mystery truth table is true in **both** the first **and** fourth valuations. While neither of these valuation sentences alone fit that pattern, they can act as parts of a larger sentence which is **true whenever one of its parts is true**.

That describes the truth conditions for a **disjunction**. Hence a disjunction of the two valuation sentences will match 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**

Call such a disjunction of valuation sentences a **valuation disjunction**. Then we see the following.

> Any truth table true in a single valuation has a matching **valuation sentence**; and a truth table true in more than one valuation has a matching **valuation disjunction**.

A larger, eight-valuation truth table illustrates. The three true valuations are matched with valuation sentences.

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	

And these three valuation sentences are combined into a valuation disjunction.

(((<u>(P ∧ Q) ∧ R</u>) ∨ (<u>(P ∧ ~Q) ∧ R</u>)) ∨ (<u>(~P ∧ Q) ∧ ~R</u>))

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	$((P \wedge Q) \wedge R)$	$((P \wedge {\sim}Q) \wedge R)$	$(({\sim}P \wedge Q) \wedge {\sim}R)$	
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 short of the general procedure we seek. If a mystery truth table is true in a single 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 sort of sentence are any help here, since they can never be false in **every** valuation. (In semantic jargon: every valuation sentence and valuation disjunction is **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're 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, "$(P \wedge {\sim}P)$". Since "$(P \wedge {\sim}P)$" is a sentence of the Chapter Two language false in every valuation, we succeed in finding a matching Chapter Two sentence for the mystery truth table.

Our general method for matching a Chapter Two sentence to each truth table then runs 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 sentence.

The second, more traditional strategy involves relaxing the 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**.[1] 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 is false in every valuation of its truth table – which is exactly the 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 contradiction, the entire disjunction will itself be a contradiction – for instance, "((P ∧ ~P) ∨ (Q ∧ ~Q))".

[1] From 2.26 §1.

Basics, basic conjunctions, and disjunctions of them, form the family of sentences said to be in **Disjunctive Normal Form** ("**DNF**" for short). The following construction rules offer precise conditions for being a DNF sentence.[2]

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, valuations disjunctions – and further sentences falling into none of those categories.

Yet despite the greater sentence-building power DNF offers over the earlier method of valuation disjunctions, such excess is largely irrelevant to our purposes. Since the sentences used in the first method – valuation sentences, valuation disjunctions, and contradictions such as "(P ∧ ~P)" – all qualify as DNF sentences, the procedure for finding such a DNF sentence remains unchanged.

[2] The construction rules for DNF sentences can be summed up quite simply in terms of **scope**: in DNF a vel has wider scope than any wedge, and a tilde has narrower scope than any wedge or vel.

- 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 – then chops that 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’re guaranteed that no truth table lies out of the reach of the Chapter Two language – the language of {~, ∧, ∨}, plus sentence letters. The formal language of Chapter Two is thus **expressively adequate**.

4. Conjunctive Normal Form. DNF impose a **hierarchy of scope** on our three connectives: negation has only a sentence letter as its scope sentence; conjunction takes only basics as its scope; and a disjunction, with the widest scope of the three, has scope over both basics and conjunctions of them.

But switching conjunctions and disjunctions in that hierarchy yields a different family of sentences, in **Conjunctive Normal Form** (**CNF**).[3] CNF embeds basics within disjunctions, and basics or disjunctions of them within conjunctions.

[3] A language is a **normal form** if it restricts negation scope to sentence letters.

Basics:

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

Basic Disjunctions:

1. Basics are basic disjunctions.
2. If ● and ▲ are basic disjunctions,
   then (●∨ ▲) is a basic disjunction.

Sentences in Conjunctive Normal Form (CNF):

1. Basic Disjunctions are CNF sentences
2. If ● and ▲ are CNF sentences,
   then (●∧ ▲) is a CNF sentence.

The basics and basic disjunctions of old are in CNF; so the following are all CNF sentences.

P	~P	(P ∨ Q)
(P ∨ ~P)	(P ∨ (~P ∨ ~Q))	(~P ∨ (~Q ∨ R))

Note that all these qualify as DNF sentences. Moreover, the basic conjunctions (however-many-barreled conjunctions of basics) found in DNF also qualify as CNF sentences (again counting basics as mutant, one-part disjunctions). So all of the following are CNF sentences.

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

CNF and DNF part company when sentences have both wedges and vels. The following sentence, for example, qualifies as CNF but not DNF.

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

Basics, basic conjunctions, and basic disjunctions thus form the common core of these two different families of sentences.

And just as every truth table is guaranteed a matching DNF sentence, so CNF likewise covers every possible truth table. Hence CNF too is expressively adequate. Since every CNF sentence is a sentence of the Chapter Two language, CNF provides a second proof that the Chapter Two formal language is expressively adequate.