4 Truth-Functional Logic

1. Introduction

In presenting a logic, the customary procedure involves four steps.

(1) specify the *syntax* of the underlying formal language, $\mathcal{L}$, over which the logic is defined;
(2) specify the *semantics* for $\mathcal{L}$, in virtue of which *semantic entailment* is defined;
(3) specify a *deductive system* for $\mathcal{L}$, in virtue of which *deductive entailment* is defined;
(4) show that semantic entailment and deductive entailment are *mutually consistent*.

In the present chapter, we discuss steps 1 and 2 for classical sentential logic.

2. The Language of Classical Sentential Logic

Classical sentential logic (CSL) can be formulated in a ZOL, either in prefix (Polish) format or in infix (algebraic) format. In the former case, the formal language, $\mathcal{L}_1$, is specified as follows, using minimal notation.

(0) The vocabulary consists of the following: p, ♯, N, K, D, C, B.
(a1) p is an atomic formula.
(a2) if σ is an atomic formula, then so is σ♯.
(a3) nothing else is an atomic formula.
(f1) every atomic formula is a formula.
(f2) if σ is a formula, then so is Nσ.
(f3) if σ_1 and σ_2 are formulas, then so are:
$K\sigma_1\sigma_2$
$D\sigma_1\sigma_2$
$C\sigma_1\sigma_2$
$B\sigma_1\sigma_2$
(f4) nothing else is a formula.

The prefix connectives correspond to negation (N), conjunction (K), disjunction (D), conditional (C), and biconditional (B), respectively.

On the other hand, the infix formulation of the language of CSL is given by formal language $\mathcal{L}_2$, which is specified as follows.

(0) The vocabulary consists of the following: P, #, $\sim$, &, $\vee$, $\rightarrow$, $\leftrightarrow$, (,).
(a1) P is an atomic formula.
(a2) if σ is an atomic formula, then so is σ♯.
(a3) nothing else is an atomic formula.
(f1) every atomic formula is a formula.
(f2) if σ is a formula, then so is $\sim\sigma$.
(f3) if σ_1 and σ_2 are formulas, then so are:
$(\sigma_1\&\sigma_2)$
$(\sigma_1\vee\sigma_2)$
$(\sigma_1\rightarrow\sigma_2)$
$(\sigma_1\leftrightarrow\sigma_2)$
(f4) nothing else is a formula.

Notice that we have employed minimal notation in the metalanguage, rather than the grammatically more explicit quote/plus notation. In particular, rather than use quotes, we simply use the very same

symbol in the metalanguage as the name of the symbol in the object language (one symbol, two meanings). Also, rather than use '+', we adopt the implicit juxtaposition method for denoting complex expressions (strings) of the object language.

As a further notational simplification, from now on, we adopt the following official metalinguistic definitions.

$p_0 =_{df} p$	$P_0 =_{df} P$
$p_1 =_{df} p\#$	$P_1 =_{df} P\#$
$p_2 =_{df} p\#\#$	$P_2 =_{df} P\#\#$
etc.	etc.

Notice that the numerical subscript is short for the number of occurrences of the sharp sign.

What's more, we will further adopt the following *informal* definitions of the customary atomic formulas of elementary logic.

$P =_{df} P_0$
$Q =_{df} P_1$
$R =_{df} P_2$
$S =_{df} P_3$
etc.[?]

[Alternatively, we could officially include all upper case Roman letters in our vocabulary, and declare that each of them is an atomic formula.]

3. Truth-Values

Ordinary sentential logic is not concerned with all sentences, but only *declarative sentences*, thus ignoring interrogative, imperative, exclamatory, and performative sentences. The simplest definition of a declarative sentence is that it is a sentence that is capable of being true or false. Basically, a declarative sentence is intended, when uttered, to declare something, which *in turn* is either true or false. It is furthermore customary to say that the sentence itself is true (resp., false) when what it declares is true (resp., false).

Associated with the adjectives 'true' and 'false' are the abstract proper nouns 'True' and 'False', which refer to what are known as *truth-values* [more about reference later].

An analogy might be useful here. Consider the difference between the adjective 'blue' and the proper noun 'Blue', as used in the following two sentences

my favorite shirt is blue

my favorite color is Blue

Observe that we capitalize the noun, in a Germanesque fashion, in order to further distinguish it from its corresponding adjective. On the other hand, we don't adopt Germanesque ontological sentiments; in particular, we do not automatically assume that there really is a *thing* (abstract or otherwise) to which the proper noun 'Blue' refers. Rather, we allow (but don't require) that the nominal use of 'blue' is merely a grammatical convenience.

In order to hear the difference between the adjectival and nominal uses of ‘blue’, it is useful to see what happens when we invert the above sentences.

blue is my favorite shirt

Blue is my favorite color

The first one sounds funny (poetic, if you like); the second one sounds rather ordinary (prosaic, if you like).

Veterans of elementary logic can render the distinction in the starkest terms, by symbolizing the two sentences, as follows.

B[s(i)]

c(i) = b

B[α]	:	α is blue
s(α)	:	α’s the favorite shirt
c(α)	:	α’s the favorite color
i	:	I/me/my
b	:	Blue

Notice also that there is a natural semantic correspondence between the adjective ‘blue’ and the noun ‘Blue’, given as follows.

object x is blue	B[x]
if and only if	↔
the color of object x is Blue	c(x) = b

Notice that this is not a logical truth (at least, not according to “standard” logic). On the other hand, it is analytically true, which is to say it is true in virtue of the meanings of its terms.

Now back to truth-values. Just as there is a conceptual relation between ‘blue’ and ‘Blue’, there is a relation between ‘true’ and ‘True’, and between ‘false’ and ‘False’. This is given as follows.

sentence $\mathbb{S}$ is true/false

if and only if

the truth-value of $\mathbb{S}$ <u>is</u> True/False

4. Truth-Functions

The customary semantics for CSL employs the notion of truth-function, which is a function that takes truth-values as input and yields truth-values as output. Formally stated:

> Df
> A *truth-function* is, by definition, an n-place function on {T,F}, for some number n.

For an account of functions, see the appendix on set theory. Basically, an n-place truth-function takes a n-tuple of truth-values as input and delivers a truth-value as output; for example, a 2-place truth-function takes a 2-tuple (ordered pair) of truth-values and delivers a truth value as output.

The following are examples of 1-place, 2-place, and 3-place truth-functions.

(e1.1) $f_1(T) = T$ [f_1 assigns T to T]
$f_1(F) = F$ [f_1 assigns F to F]

(e1.2) $f_2(T) = F$
$f_2(F) = T$

(e1.3) $f_3(T) = T$
$f_3(F) = T$

(e1.4) $f_4(T) = F$
$f_4(F) = F$

(e2.1) $g_1(T,T) = T$
$g_1(T,F) = F$
$g_1(F,T) = F$
$g_1(F,F) = F$

(e2.2) $g_2(T,T) = F$
$g_2(T,F) = T$
$g_2(F,T) = T$
$g_2(F,F) = T$

(e3.1) $h_1(T,T,T) = T$
$h_1(T,T,F) = T$
$h_1(T,F,T) = F$
$h_1(T,F,F) = F$
$h_1(F,T,T) = T$
$h_1(F,T,F) = F$
$h_1(F,F,T) = T$
$h_1(F,F,F) = F$

(e3.2) $h_2(T,T,T) = T$
$h_2(T,T,F) = F$
$h_2(T,F,T) = F$
$h_2(T,F,F) = T$
$h_2(F,T,T) = T$
$h_2(F,T,F) = F$
$h_2(F,F,T) = T$
$h_2(F,F,F) = F$

How many truth-functions are there. Standard combinatorial reasoning yields the following finite results (1)-(n). Set theory yields the general result.

(1)	The number of 1-place truth-functions:	4	
(2)	The number of 2-place truth-functions:	16	
(3)	The number of 3-place truth-functions:	256	
(4)	The number of 4-place truth-functions:	64k	[k = 1024]
(5)	The number of 5-place truth-functions:	4096m	[$m = k^2$]
(n)	The number of n-place truth-functions:	2 exp (2 exp n)	
(g)	The number of truth-functions:	infinitely-many	

For example, in (2) there are four 2-tuples of truth-values; each one can be assigned T or F; so for each 2-tuple there are 2 possible assignments. Accordingly, the total number of possible assignments is $2\times2\times2\times2$, which is 16. In the case of an n-place truth-function, there are 2^n (i.e., 2 exp n) different n-tuples; for each n-tuple, there are 2 possible assignments, so the total number of possible assignments is 2 exp (2 exp n). The latter can be quite large.

5. Truth-Functional Semantics for CSL

Intimately related to truth-functions are truth-functional connectives. A connective is not in and of itself truth-functional, but is truth-functional only relative to a semantics. A semantics for a formal language $\mathcal{L}$ provides, at the minimum, a set of *admissible valuations* on $\mathcal{L}$, which are defined as follows.

Df

Let $\mathcal{L}$ be a language, and let $S(\mathcal{L})$ be the set of sentences (formulas) of $\mathcal{L}$. Then a *valuation* on $\mathcal{L}$ is any function from $S(\mathcal{L})$ into {T,F}. A *truth-value semantics* on $\mathcal{L}$ is, by definition, any set of valuations on $\mathcal{L}$.

In this context, let us drop the prefix 'truth-value', and simply refer to a set V of valuations as a semantics.

We are now in a position to define truth-functionality.

Df

Let χ be an n-place connective in a prefix-formatted language $\mathcal{L}$. Let V be a (truth-value) semantics for $\mathcal{L}$. Then χ is *truth-functional* relative to V iff: there is an n-place truth-function, call it f_χ, such that, for every valuation v in V, for any formulas $\phi_1,...,\phi_n$,

$$v(\chi\phi_1...\phi_n) = f_\chi(v(\phi_1),...,v(\phi_n))$$

The basic idea is simple; a connective is truth-functional iff it corresponds to a truth-function. Insofar as connective χ corresponds to truth-function f_χ, the truth-value of any χ-formula is a function (specifically, f_χ) of the respective truth-values of its constituents.

If every connective of $\mathcal{L}$ is truth-functional relative to V, we say that V is a truth-functional semantics for $\mathcal{L}$. This is made official in the following.

Df

Let $\mathcal{L}$ be a ZOL, and let V be a semantics for $\mathcal{L}$. Then V is *truth-functional* iff every connective χ of $\mathcal{L}$ is truth-functional relative to V.

The usual semantics for CSL is truth-functional. The following is a semi-formal definition of this semantics, for the prefix-formatted language $\mathcal{L}_1$.

Df

The usual semantics for CSL, in prefix-format, countenances as admissible all and only those valuations on $\mathcal{L}_1$ that satisfy the following restrictions.

(N) $v(N\alpha) = n(v(\alpha))$
(K) $v(K\alpha\beta) = k(v(\alpha),v(\beta))$
(D) $v(D\alpha\beta) = d(v(\alpha),v(\beta))$
(C) $v(C\alpha\beta) = c(v(\alpha),v(\beta))$
(B) $v(B\alpha\beta) = b(v(\alpha),v(\beta))$

Here, the truth-functions are defined as follows.

(n) n(T)=F; n(F)=T
(k) k(T,T)=T; k(T,F)=F; k(F,T)=F; k(F,F)=F
(d) d(T,T)=T; d(T,F)=T; d(F,T)=T; d(F,F)=F
(c) c(T,T)=T; c(T,F)=F; c(F,T)=T; d(F,F)=T
(b) b(T,T)=T; b(T,F)=F; b(F,T)=F; b(F,F)=T

The functions n,k,d,c,b are of course the familiar truth-functions associated, respectively, with negation, conjunction, disjunction, conditional, and biconditional. For example, the fact that k(T,T)=T amounts to the fact that the "conjunction" of T and T is T.

Whether we actually call the function k conjunction depends upon how precise we wish to be. If we insist that conjunction is a connective, then the function k is not conjunction, since it is not a connective; rather, k is the truth-function that corresponds to conjunction. Of course, in intro logic, the connective and the truth-function were both called conjunction. [Intro students have enough trouble without having to worry about the distinction between (set theoretic) functions and (syntactic) functors.]

On the other hand, it is convenient (if somewhat sloppy) to use the term 'conjunction' to refer to both the functor K and the function k. This allows us to describe the truth conditions for the functor K as follows.

(t) the truth-value of the conjunction of two formulas is the conjunction of the truth-values of the two formulas.

The latter statement can be made more precise, if we distinguish between *syntactic conjunction* and *semantic conjunction*, in which case (t) is rewritten as follows.

(t*) the truth-value of the *syntactic conjunction* of two formulas is the *semantic conjunction* of the truth-values of the two formulas.

Writing both "conjunctions" in infix notation, and using the same symbol '&' for both, we can re-write (t) as follows.

(t**) $v(\alpha\&\beta) = v(\alpha)\ \&\ v(\beta)$

Here, '&' is ambiguous: the first occurrence of '&' is the name of the ampersand symbol of the object language; the second occurrence is the name of the truth-function k, which is a set of ordered pairs. The difference between syntactic and semantic conjunction is striking; whereas $(\alpha\&\beta)$ is a string consisting of '(' followed by α followed by '&' followed by β followed by ')', $v(\alpha)\&v(\beta)$ is not a string but a truth-value; for example, T&T is not a string consisting of T followed by & followed by T; T&T is just T [T&T=T].

The usual semantics for CSL is truth-functional. A simple example of a non-truth-functional semantics for $\mathcal{L}_1$ is easy to construct.

(D) The semantics NTFS for $\mathcal{L}_1$ countenances as admissible exactly one valuation, namely w defined as follows.

$w(\alpha)=T$ if $v(\alpha)=T$ for every $v \in V(TFS)$; $v(\alpha)=F$, otherwise.

Here, V(TFS) is the set of admissible valuations of the usual truth-functional semantics, mentioned above.

In other words, the valuation w assigns T to all tautologies of ordinary classical SL, but F to all non-tautologies.

To show that NTFS is not truth-functional, we need merely show that one connective is not truth-functional. Consider negation; first, consider the formula Np; the input formula p is not a tautology of classical logic, so p is false in NTSF; similarly, the output formula Np is not a tautology, so Np is also

false in NTSF. Input: false; output: false. Now, consider the negation NKpNp; the input formula KpNp is not a tautology of CL, so it is false in NTSF; on the other hand, NKpNp is a tautology of CL, so the output formula NKpNp is true in NTSF; input: false; output: true. Thus, relative to this semantics, the truth-value of a negation is not a function of the truth-values of its constituents.

In order to produce a somewhat more interesting example of a non-truth-functional semantics, let us first enlarge $\mathcal{L}_1$ by adding the one-place connective 'L'; then let us modify the semantics TFS, to produce TFS+, by adding the following clause to the definition of admissible valuation.

(L) $v(L\alpha) = T$ if $\forall w(w \in V \rightarrow w(\alpha)=T)$; $v(L\alpha) = F$ otherwise.

'L' corresponds roughly to the English "it is necessary that..."; $L\alpha$ is true if α is true in every valuation, and is false otherwise.

To see that the added connective L is not truth-functional relative to TFS+, consider the formulas LDpNp and Lp. Since p is atomic, some valuations make p true and others make p false; since some valuations make p false, every valuation makes Lp false. On the other hand, every valuation makes DpNp true, so every valuation makes LDpNp true. Consider a valuation, call it v, that makes p true; then v(p)=v(DpNp)=T, but v(Lp)=F, and v(LDpNp)=T. Thus the truth-value of $L\alpha$ is not a function of the truth-value of α.

6. Expressive Completeness

Ordinary classical sentential logic (CSL) employs only five connectives, so the semantics of CSL only involves five truth-functions. Yet there are infinitely many truth-functions, and hence there are (in principle) infinitely many truth-functional connectives. As you already know from intro logic, many "non-standard" truth-functional connectives can be paraphrased using "standard" truth-functional connectives. For example, 'neither...nor' sentences can be paraphrased using 'not' and 'and'; specifically, 'neither P nor Q' may be paraphrased as 'not-P and not-Q'.

The obvious question that arises is whether every truth-functional connective (explicit or otherwise) can be paraphrased using standard connectives. If the answer is 'yes', then the standard connectives are *expressively complete*; if the answer is 'no', then the standard connectives are *expressively incomplete*.

In formalizing this idea, we present the following definitions. Note carefully: In what follows, we presuppose a ZOL $\mathcal{L}$ and a semantics $\mathcal{V}$ for $\mathcal{L}$; all definitions are relative to $\mathcal{L}$ and $\mathcal{V}$.

Df

Two formulas α and β are said to be *semantically equivalent* iff $v(\alpha)=v(\beta)$ for every admissible valuation.

Df

Let $\mathbb{C}$ be a collection of connectives, and let χ be a connective. Then χ is *expressible in terms of* $\mathbb{C}$ iff every formula involving χ is semantically equivalent to a formula involving just the connectives in $\mathbb{C}$.

Examples:

Relative to the usual semantics for CSL, we have the following.

(1) & is expressible in terms of $\{\vee,\sim\}$, and $\{\rightarrow,\sim\}$

(2) $\vee$ is expressible in terms of $\{\&,\sim\}$, and $\{\rightarrow,\sim\}$

(3) $\rightarrow$ is expressible in terms of $\{\vee,\sim\}$, and $\{\&,\sim\}$

(4) $\leftrightarrow$ is expressible in terms of $\{\vee,\sim\}$, and $\{\&,\sim\}$, and $\{\rightarrow,\sim\}$

(5) $\sim$ is ***not*** expressible in terms of $\{\rightarrow,\leftrightarrow,\vee,\&\}$

(6) $\&,\vee,\rightarrow$ are ***not*** expressible in terms of $\{\leftrightarrow,\sim\}$

For example, & is expressible in terms of $\{\vee,\sim\}$ since $(\alpha\&\beta)$ is semantically equivalent to $\sim(\sim\alpha\vee\sim\beta)$ for any formulas α, β.

Our next definition is a modification of the previous definition, in which we substitute 'truth-function' for 'connective'. It takes into account that not every truth-function need be explicitly captured in a given language.

Df

Let $\mathbb{C}$ be a collection of connectives, and let f be an n-place truth-function. Then f is *expressible in terms* of $\mathbb{C}$ iff there is a formula $\mathbb{F}[P_1,...,P_n]$ involving n atomic formulas $P_1,...,P_n$ and just the connectives of $\mathbb{C}$ such that for every admissible valuation v,

$$v(\mathbb{F}[P_1,...,P_n]) = f(v(P_1),...,v(P_n))$$

We can now give the general definition of expressive completeness.

Df

A collection $\mathbb{C}$ of connectives is *expressively complete* iff every truth-function is expressible in terms of $\mathbb{C}$.

In what follows, we argue that every truth-function is expressible in terms of $\{\sim, \vee, \&\}$. Consider an arbitrary n-place truth-function, call it f. There are 2 cases to consider; Case 1: f is a constant function $f(i)=F$ for every possible input; Case 2: f assigns T to at least one input. The first case is trivial; any contradiction $(P\&\sim P)$ expresses the truth-function. So let's move to Case 2. By hypothesis, f is an n-place function, so there are 2^n possible input; enumerate these. By doing so, we have in effect the guide table for a truth table. Now, go through the enumeration (truth table) as follows. Consider the input $(v_1,...,v_n)$ truth-values; if $f(v_1,...,v_n) = F$, then skip this item; on the other hand, if $f(v_1,...,v_n) = T$, then write down the following sequence of formulas:

$$\phi_1, \ldots, \phi_n$$

where

$$\phi_i = P_i \quad \text{if} \quad v_i=T$$
$$\phi_i = \sim P_i \quad \text{if} \quad v_i=F$$

Here, P_i is the ith atomic formula. Next, take the conjunction of the resulting sequence $(\phi_1,...,\phi_n)$ of formulas. Then go to the next line in the truth table. Having gone through the truth table, take all the resulting conjunctions, and form their disjunction. Claim: the resulting disjunction of conjunctions is a formula whose truth table corresponds to the function f. (The proof of the latter claim is left as an exercise.)

Terminology: a formula constructed in the above manner is said to be in *disjunctive normal form* (DNF).

At this point, let us do a few examples, to see how the DNF technique works. Consider the following 2-place truth-functions.

$f(T,T) = F$
$f(T,F) = F$
$f(F,T) = T$
$f(F,F) = F$

The corresponding DNF formula is: $(\sim P_1 \& P_2)$

$f(T,T) = T$
$f(T,F) = F$
$f(F,T) = T$
$f(F,F) = F$

The corresponding DNF formula is: $(P_1 \& P_2) \vee (\sim P_1 \& P_2)$

$f(T,T) = T$
$f(T,F) = T$
$f(F,T) = T$
$f(F,F) = F$

The corresponding DNF formula is: $(P_1 \& P_2) \vee (P_1 \& \sim P_2) \vee (\sim P_1 \& P_2)$

Next, consider the following 3-place functions.

f(T,T,T) = T	f(F,T,T) = T
f(T,T,F) = T	f(F,T,F) = F
f(T,F,T) = F	f(F,F,T) = T
f(T,F,F) = F	f(F,F,F) = F

The corresponding DNF formula is:

$$(P_1 \,\&\, P_2 \,\&\, P_3) \vee (P_1 \,\&\, P_2 \,\&\, \sim P_3) \vee (\sim P_1 \,\&\, P_2 \,\&\, P_3) \vee (\sim P_1 \,\&\, \sim P_2 \,\&\, P_3).$$

f(T,T,T) = T	f(F,T,T) = T
f(T,T,F) = F	f(F,T,F) = T
f(T,F,T) = T	f(F,F,T) = F
f(T,F,F) = F	f(F,F,F) = F

The corresponding DNF formula is:

$$(P_1 \,\&\, P_2 \,\&\, P_3) \vee (P_1 \,\&\, \sim P_2 \,\&\, P_3) \vee (\sim P_1 \,\&\, P_2 \,\&\, P_3) \vee (\sim P_1 \,\&\, P_2 \,\&\, \sim P_3).$$

Finally, we observe that the connectives $\sim$, &, $\vee$ can all be expressed in terms of a single connective "nor", which corresponds to 'neither...nor' (exercise). Similarly, they are expressible in terms of "nand", which corresponds to 'not both...and...'. Given the earlier theorem, it follows that every truth-function is expressible in terms of a single truth-functional connective.

7. Exercises

1. Truth-Functions

Define function; define truth-function; give examples from English of a 1-place, a 2-place, and a 3-place, truth-functional connective. In each case, write down the corresponding truth function in 'f(a)=v' notation.

2. Expressing Connectives in Terms of Each Other

a. Express the five standard truth-functional connectives in terms of '$\sim$' and '&'.

b. Express the five standard truth-functional connectives in terms of '$\sim$' and '$\vee$'.

c. Express the five standard truth-functional connectives in terms of '$\sim$' and '$\rightarrow$'.

d. Express the five standard truth-functional connectives in terms of the "nor" connective '$\downarrow$', which corresponds to 'neither...nor...'.

e. Express the five standard truth-functional connectives in terms of the "nand" connective '$\uparrow$', which corresponds to 'not both...and...'

3. Disjunctive Normal Form

Convert each of the following formulas into canonical disjunctive normal form; in other words, first construct the associated n-place truth function, then write down the DNF formula that yields this truth function. You may use 'P', 'Q', 'R', etc. in place of 'P_0', 'P_1', 'P_2', etc.

$P \rightarrow Q$; $P \vee Q$; $P \leftrightarrow Q$; $P \rightarrow (Q \& R)$; $P \& (Q \vee R)$

8. Answers to Selected Exercises

2a.

a.

P∨Q	≡	∼(∼P&∼Q)
P→Q	≡	∼(P&∼Q)
P↔Q	≡	∼(P&∼Q)&∼(Q&∼P)

b.

P&Q	≡	∼(∼P∨∼Q)
P→Q	≡	∼P∨Q
P↔Q	≡	∼[∼(∼P∨Q)∨∼(∼Q∨P)]

c.

P&Q	≡	∼(P→∼Q)
P∨Q	≡	∼P→Q
P↔Q	≡	∼[(P→Q)→∼(Q→P)]

d.

∼P	≡	P↓P
P∨Q	≡	(P↓Q)↓(P↓Q)
P&Q	≡	(P↓P)↓(Q↓Q)
P→Q	≡	[(P↓P)↓Q]↓[(P↓P)↓Q]
P↔Q	≡	{[(P↓P)↓(Q↓Q)]↓(P↓Q)}↓{[(P↓P)↓(Q↓Q)]↓(P↓Q)}

e.

∼P	≡	P↑P
P&Q	≡	(P↑Q)↑(P↑Q)
P∨Q	≡	(P↑P)↑(Q↑Q)
P→Q	≡	P↑(Q↑Q)
P↔Q	≡	{[P↑(Q↑Q)]↑[Q↑(P↑P)]}↑{[P↑(Q↑Q)]↑[Q↑(P↑P)]}

3.

P→Q	≡	(P&Q) ∨ (∼P&Q) ∨ (∼P&∼Q)
P∨Q	≡	(P&Q) ∨ (P&∼Q) ∨ (∼P&Q)
P↔Q	≡	(P&Q) ∨ (∼P&∼Q)
P→(Q&R)	≡	(P&Q&R)∨(∼P&Q&R)∨(∼P&Q&∼R) ∨ (∼P&∼Q&R)∨(∼P&∼Q&∼R)
P&(Q∨R)	≡	(P&Q&R) ∨ (P&Q&∼R) ∨ (P&∼Q&R)