3.9. Nested Conditional Form

1. Nested Conditionals. We established already that the language {→, ~} is expressively adequate, because it is provably equivalent to a language such as {∧, ~} which is known to be expressively adequate. But that argument didn't provide the sort of general procedure we had in DNF, for matching each truth table with a corresponding sentence. Here we return to the {→, ~} language and consider how to build a {→, ~} sentence for any given truth table. Central to this task will be understanding a nested conditional matching a series of sentences.

A **series of sentences** is just a list of sentences, in a certain order. (We show the intended order in the usual way: starting from the left, we list the first sentence in the series, the second, and so on.) The following is a simple example.

~P, (P ∧ ~Q), (R ∨ S)

Copyright Brian Beakley 2017

Corresponding to such a series of sentence letters we build a **nested conditional**: a conditional containing a smaller conditional as consequent, where that smaller conditional may itself contain a (yet) smaller conditional as consequent, and so on.

We build a nested conditional out of a series of sentences by working our way backwards through the list. We first building a conditional with the **last sentence as consequent** and **next-to-last letter** as **antecedent**. So from the above series of three sentences we begin like this.

((P ∧ ~Q) → (R ∨ S))

If there is a yet earlier sentence in the series – here, "~P" – we build a conditional with the previous conditional as consequent and this sentence as antecedent.

(~P → ((P ∧ ~Q) → (R ∨ S)))

After doing this as many times as needed to include each sentence in the series, we have a nested conditional that correspond to the original series of sentences.

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

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

And while such a series can contain any sort of formal sentence from our current formal language, we will be especially interested in series of **sentence letters**, listed in alphabetical order[1] – for example, the following.

P, Q, R, S

Corresponding to such a series of sentence letters we build a nested conditional. So the above series of sentence letters yields this nested conditional.

$(P \rightarrow (Q \rightarrow (R \rightarrow S))$

So a nested conditional of sentence letters corresponds to a series of sentence letters in the following general pattern.

> A **series of sentence letters** is a list of sentence letter listed in alphabetical order (with the first sentence listed first, i.e., on the left; followed by the second letter, and so).
>
> $\bullet_{\mathbf{1}}, \ldots, \bullet_{\mathbf{k}}$ (for **k** many sentence letters)

[1] To be completely general: sentence letters are listed in the order "P", "Q" … "Z", "P_1" "Q_1" … "Z_1", "P_2", etc.

A **nested conditional** corresponding to this series is a conditional with the first letter in the list, $\bullet_1$, as antecedent, and as consequent a smaller conditional itself having the second letter in the list, $\bullet_2$, as antecedent, and so on, ending with a conditional having the next-to-last sentence $\bullet_{k-1}$ as antecedent and last sentence $\bullet_k$ as consequent.

$$(\bullet_1 \rightarrow (\ldots (\bullet_{k-1} \rightarrow \bullet_k) \ldots))$$

We will then modify this nested conditional by inserting tildes before some sentence letters. Our goal will be to replicate the behavior of valuation and counter-valuation sentences, but phrased entirely in the language $\{\rightarrow, \sim\}$.

Recall that a **valuation sentence** is a sentence in the language $\{\sim, \wedge\}$) true in exactly one valuation, while a **counter-valuation sentence** is a sentence of the $\{\sim, \vee\}$ language false in just one valuation. [2] To construct a $\{\rightarrow, \sim\}$ counterpart of a counter-valuation sentence, we first build a nested conditional featuring just the sentence letters listed in that valuation. So for the following valuation we begin with this nested conditional.

Valuation:

P	**Q**	**R**	**S**
1	0	1	1

$(P \rightarrow (Q \rightarrow (R \rightarrow S)))$

[2] Discussed in 2.26 and 2.27. Note that **semantically** conditionals are more like disjunctions than like conjunctions, because in the semantics both the disjunction and conditional rules are true in three out of the four lines. The result, with disjunctions, was a counter-valuation sentence (built from a family of sentence letters) false in only one valuation. So we'll find it easiest here to replicate that result with conditions and negations (and then simulate a valuation sentence by adding a tilde).

Next we **add tildes** according to following rule.

> For **all but the last sentence letter**: add a **tilde** to that sentence letter if (and only if) that letter is **false** in the given valuation.
>
> For the **last sentence letter**: add a **tilde** if (and only if) that letter is **true** in the given valuation.

So we modify the nested conditional by adding tildes: to "Q" (because it's false in this valuation) and to "S" (because it's the last letter and is true in this valuation).

Valuation:

P	**Q**	**R**	**S**
1	0	1	1

(P → (~Q → (R → ~S)))

We know from the semantic rule for conditionals that this sentence will be false in only one sort of valuation: where the antecedent **"P" is true** but the consequent is false. And the only valuation where the consequent "(~Q → (R → ~S)) is false is one where "~Q" is true (hence where **"Q" is false**) and the consequent "(R → ~S)" is false. Finally, "(R → ~S)" is false only where its antecedent **"R" is true** and its consequent "~S" is false (hence where **"S" is true**).

Summing up: "**(P → (~Q → (R → ~S)))**" is **false only where "P" is true, "Q" is false, "R" is true, and "S" is true**. But that's just the valuation we started with. So by adding tildes to a nested conditional we get a conditional false in exactly the valuation given. We thus have a general recipe for building the {→, ~} counterpart to a counter-valuation sentence – call it a "**counter-valuation conditional**".

> **Counter-Valuation Conditional**: a nested conditional false in exactly one valuation.

And since such a sentence is false only in the valuation given, its **negation** will be true only in that valuation. So from a given valuation we can build the negation of a nested conditional which is true in just that valuation. Call such a negation of a counter-valuation conditional a "**valuation negation**".

> **Valuation Negation**: the negation of a counter-valuation conditional (hence: the negation of a nested conditional, which is true in exactly one valuation)

So "**~(P → (~Q → (R → ~S)))**" is **true only where "P" is true, "Q" is false, "R" is true, and "S" is true**.

And of course, beginning with a valuation negation we can always work backwards to see which valuation makes it true, by modifying the earlier tilde-inserting rules.

> For **all but the last sentence letter**: that letter is **false** in the given valuation if it has a **tilde** before it (and true in that valuation if has no tilde).
>
> For the **last sentence letter**: that letter is **true** in the given valuation if it has a **tilde** before it (and false in that valuation if has no tilde)

For instance, the sentence "**~(~P → (Q → (R → (~S → ~T))))**" is (in a 32-valuation truth table for "P" through "T") true in just the following valuation.

P	Q	R	S	T
0	1	1	0	1

2. Nested Conditional Form. Recall that once we had valuation sentences in hand in Chapter Two, we could match any single valuation with a corresponding {~, ∧} sentence (namely, a valuation sentence). But for a sentence true in more than one valuation we proceeded to build a **disjunction of valuation sentences** (one valuation sentence matching each valuation where that sentence is true) – resulting in Disjunctive Normal Form (DNF). An equivalent move, performed in a manner

not taking us outside the {~, →} language, calls for a {~, →} equivalent of a disjunction.

That is easily come by – for "(P ∨ Q)" is equivalent to "(~P → Q)," "(P ∨ (Q ∨ R))" is equivalent to "(~P → (~Q → R))," and so on. In general: for a set of (two or more) valuation negations {~▲$_1$, …, ~▲$_M$}, each true in exactly one valuation, we can build a nested conditional of the general form (▲$_1$ → (▲$_2$ → … (▲$_{M-1}$ → ~▲$_M$) …).[3]

For example, the following truth table makes a (mystery) sentence true in just the first and last valuations.

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

For the first valuation we build the valuation negation "~(P → ~Q)", and for the fourth the valuation negation "~(~P → Q)". To disjoin them together we build the nested conditional "(~~(P → ~Q) → ~(~P → Q))". Clearing the double negation from the antecedent leaves "((P → ~Q) → ~(~P → Q))". This sentence matches the truth table.

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

[3] Since the valuation negations {~▲$_1$, …, ~▲$_M$} already begin with a tilde, inserting these negations into a sentence of the form "(~●$_1$ → (… → (~●$_{N-1}$ → ●$_N$)…)" (where all of the antecedents are negated) yields a nested conditional of the form "(~~▲$_1$ → (… → (~~▲$_{M-1}$ → ~▲$_M$)…)". Eliminating double negations on the antecedents yields the simplified form listed above: "(▲$_1$ → (… → (▲$_{M-1}$ → ~▲$_M$)…)".

A sentence true in one valuation is thus equivalent to a valuation negation, and a sentence true in more than one valuation is equivalent to a nested conditional of valuation negations and a counter-valuation conditional. We round out this list by assigning ~(P → P) to the truth table false in every valuation.

Call any sentence in one of these forms a sentence in **Nested Conditional Form (NCF)**. We are guaranteed that any possible truth table will be matched by some NCF sentence; and all these sentences are in the {~, →} language. We thus have a general method for finding a {~, →} sentence for any given truth table.

Summary

Nested Conditionals:

- A **series of sentence letters** is a list of sentence letter listed in alphabetical order (with the first sentence listed first, i.e., on the left; followed by the second letter, and so).

  $\bullet_1, \ldots, \bullet_k$ (for **k** many sentence letters)

- A **nested conditional** corresponding to this series is a conditional with the first letter in the list, $\bullet_1$, as antecedent, and as consequent a smaller conditional itself having the second letter in the list, $\bullet_2$, as antecedent, and so on, ending with a conditional having the next-to-last sentence $\bullet_{k-1}$ as antecedent and last sentence $\bullet_k$ as consequent.

Counter-Valuation Conditionals: a counter-valuation conditional is a nested conditional false in exactly one valuation. From a given valuation, a counter-valuation conditional is built from a nested conditional according to the following rule.

- For **all but the last sentence letter**: add a **tilde** to that sentence letter if (and only if) that letter is **false** in the given valuation.

- For the **last sentence letter**: add a **tilde** if (and only if) that letter is **true** in the given valuation.

Valuation Negation: the **negation of a counter-valuation conditional** (hence: the negation of a nested conditional, which is true in exactly one valuation)

A sentence is in **Nested Conditional Form** (**NCF**) if it is in one of the following three forms.

- For a truth table true in **just one valuation**, the corresponding NCF sentence is the valuation negation true in that valuation.

- For a truth table true in **more than one valuation**, where each valuation is matched by a valuation negation, the corresponding NCF sentence is a nested conditional of those valuation negation built according to the following rule.

  For a set of (two or more) valuation negations $\{\sim▲_1, \ldots, \sim▲_M\}$, each true in exactly one valuation, build a nested conditional of the general form:

  $$(▲_1 \rightarrow (▲_2 \rightarrow \ldots (▲_{M-1} \rightarrow \sim▲_M) \ldots).$$

  This sentence will be true in just those valuations where the truth table is true.

- For a truth table **false in every valuation**, the corresponding NCF sentence is the negated conditional "$\sim(P \rightarrow P)$".