Proposition Logic

Robert Keller

21 February 2007

Proposition logic is the most basic of several important families of logic, and underlies many of those families. Although some of the results may seem obvious, they are presented for the purpose of better understanding more complex families, particularly predicate logic, which is discussed subsequently.

Proposition logic is concerned with relationships among formulas in which the atomic propositions have simple truth values (true, false). The formulas are inductively defined to be either atomic propositions or ones built up from other formulas using *connectives*: $\wedge$, $\rightarrow$, $\bot$.

A formula is defined by the following context-free grammar, having start symbol F.

F → A | (F ∧ F) | (F '→' F)	*formulas*
A → P | ⊥	*atomic formulas*
P → p | P'	*proposition symbols*

The reason for the quotes around on the right-hand side $\rightarrow$ is to distinguish this symbol, read *implies*, from the meta-symbol used to define productions. The symbol $\wedge$ is read *and*, and the symbol is read *bottom* or *falsum*. The productions for proposition symbols generate an unlimited number of symbols p, p', p'', … . However, it is common practice to use q, r, s, … to stand for these so that the resulting formulas or more readable.

Another technique sometimes used is to introduce *abbreviations* for certain formulas:

For any formulas φ, ψ:

$\varphi \leftrightarrow \psi$ abbreviates $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$

$\neg\varphi$ abbreviates $(\varphi \rightarrow \bot)$

$\varphi \vee \psi$ abbreviates $\neg(\neg\varphi \wedge \neg\psi)$

The purpose of introducing abbreviations as opposed to new formulas is to reduce the number of distinct types of formulas that need to be considered in various structural induction proofs.

To make formulas more readable, another type of shortcut, based on precedence or binding strength, is used to reduce the number of parentheses:

- $\neg$ binds more tightly than any other connective.
- $\wedge$ binds more tightly than any $\rightarrow$, $\leftrightarrow$, and $\vee$.
- $\vee$ binds more tightly than $\rightarrow$ and $\leftrightarrow$.
- $\rightarrow$ binds more tightly than $\leftrightarrow$.

For example $\neg p \wedge \neg q \rightarrow r \vee s$ is a shortcut for $(((\neg p) \wedge (\neg q)) \rightarrow (r \vee s))$. When the result would otherwise look ambiguous, it is better to use parentheses rather than shortcuts.

Semantics

Semantics is the endeavor of assigning meanings to formulas. Meanings are expressed of truth values and valuations, which assign a truth value to each atomic formula.

We will use {0, 1} for our truth values, interpreting 0 as *false* and 1 as *true*.

Definition A *valuation* is a map ν that assigns a truth value to every atomic formula, but with the requirement that $\nu(\bot) = 0$. In other words, a valuation is determined by a truth value for every proposition symbol.

Some examples of valuations are:

$$\nu_1 = \begin{pmatrix} p & p' & p'' & p''' & \dots \\ 0 & 1 & 0 & 1 & \end{pmatrix}$$

$$\nu_2 = \begin{pmatrix} p & p' & p'' & p''' & \dots \\ 0 & 1 & 1 & 0 & \end{pmatrix}$$

$$\nu_3 = \begin{pmatrix} p & p' & p'' & p''' & \dots \\ 1 & 1 & 1 & 1 & \end{pmatrix}$$

There is an uncountable infinity of valuations, but for any single formula, only a finite set of equivalence classes matters, where two valuations are equivalent when they assign the same truth values to each of the proposition symbols occurring in the formula.

Definition For any formula φ, the *value induced* by a valuation ν is a truth value $\nu(\varphi)$ according to the following induction:

- If φ is atomic, the induced value $\nu(\varphi)$ is just the value given by the valuation.
- If φ is of the form $(\psi \wedge \xi)$, the induced value $\nu(\varphi)$ is $H_\wedge(\nu(\psi),\nu(\xi))$, where $H_\wedge$ is the function defined below.

a	**b**	$\mathbf{H_\wedge(a, b)}$
0	0	0
0	1	0
1	0	0
1	1	1

- If φ is of the form (ψ→ ξ), the induced value ν(φ) is $H_{\rightarrow}(\nu(\psi),\nu(\xi))$, where $H_{\rightarrow}$ is the function defined below.

a	**b**	$\mathbf{H_{\rightarrow}(a, b)}$
0	0	1
0	1	1
1	0	0
1	1	1

The following can be *inferred* from the above definitions:

- If φ is of the form (¬ψ), the induced value ν(φ) is $H_{\neg}(\nu(\psi))$, where $H_{\neg}$ is the function defined below.

a	$\mathbf{H_{\neg}(a)}$
0	1
1	0

- If φ is of the form (ψ ∨ ξ), the induced value ν(φ) is $H_{\vee}(\nu(\psi),\nu(\xi))$, where $H_{\vee}$ is the function defined below.

a	**b**	$\mathbf{H_{\vee}(a, b)}$
0	0	0
0	1	1
1	1	1
1	1	1

- If φ is of the form (ψ ↔ ξ), the induced value ν(φ) is $H_{\leftrightarrow}(\nu(\psi),\nu(\xi))$, where $H_{\leftrightarrow}$ is the function defined below.

a	**b**	$\mathbf{H_{\leftrightarrow}(a, b)}$
0	0	1
0	1	0
1	1	0
1	1	1

There is good reason for introducing the connectives as merely symbols and not truth functions: there will be other uses for them. However, that does not stop us from interpreting them as truth functions when inducing a valuation. For example, the induced value of formula (((¬p) ∧ (¬q)) → (r ∨ s)) given the valuation

$$\nu = \begin{pmatrix} p & q & r & s & \dots \\ 1 & 0 & 0 & 0 & \end{pmatrix}$$

could be interpreted as evaluating (((¬1) ∧ (¬0)) → (0 ∨ 0)) = ((0 ∧ 1) → 0) = (0 → 0) = 1, when we are really evaluating $H_{\rightarrow}(H_{\wedge}(H_{\neg}(1), H_{\neg}(0)), H_{\vee}(0, 0))$.

It is good to acquire some facility with computing induced values. There are several shortcuts that can be remembered:

- The result of $H_{\wedge}$ is always 0, unless both arguments are 1.
- The result of $H_{\vee}$ is always 1, unless both arguments are 0.
- The result of $H_{\rightarrow}$ is always 1, unless the first argument is 1 and the second 0.
- The result of $H_{\leftrightarrow}$ is 1 iff both arguments have the same value.

Definition A valuation ν *satisfies* formula φ, and φ *is satisfied by* ν, iff $\nu(\varphi) = 1$.

Definition A *tautology* is a formula φ that is satisfied by every valuation ν.
Being a tautology may be expressed by the notation:

$$\models \varphi$$

One way, by no means the only, to determine whether a formula is a tautology is to use the truth table method.

Truth Table Method

To determine whether or not a formula is a tautology, enumerate all combinations of assignments of {0, 1} to the proposition symbols in the formula. Compute the induced value for each combination, and check whether or not each value is 1. The construction can stop early if a value of 0 is reached.

Example Determine whether $(p \rightarrow q) \leftrightarrow ((\neg q) \rightarrow (\neg p))$ is a tautology:

valuation class		**induced values**				
p	**q**	**$\neg$q**	**$\neg$p**	**p$\rightarrow$ q**	**($\neg$q)$\rightarrow$ ($\neg$p)**	**(p$\rightarrow$ q) $\leftrightarrow$ (($\neg$q)$\rightarrow$ ($\neg$p))**
0	0	1	1	1	1	1
0	1	0	1	1	1	1
1	0	1	0	0	0	1
1	1	0	0	1	1	1

Thus the formula is a tautology.

There are numerous tautologies worth deriving and remembering. Those involving $\leftrightarrow$ at the top level can be viewed as *logical equivalences*. Here is a partial list:

$(p \rightarrow q) \leftrightarrow ((\neg p) \vee q)$	$(\neg(p \rightarrow q)) \leftrightarrow (p \wedge (\neg q))$
$(p \rightarrow q) \leftrightarrow ((\neg q) \rightarrow (\neg p))$	$\neg\neg p \leftrightarrow p$
$(p \wedge q) \leftrightarrow (q \wedge p)$	$(p \vee q) \leftrightarrow (q \vee p)$

$(p \wedge (q \wedge r)) \leftrightarrow ((p \wedge q) \wedge r)$	$(p \vee (q \vee r)) \leftrightarrow ((p \vee q) \vee r)$
$(p \wedge (q \vee r)) \leftrightarrow ((p \wedge q) \vee (p \wedge r))$	$(p \vee (q \wedge r)) \leftrightarrow ((p \vee q) \wedge (p \vee r))$
$\neg(p \vee q) \leftrightarrow ((\neg p) \wedge (\neg q))$	$\neg(p \wedge q) \leftrightarrow ((\neg p) \vee (\neg q))$
$(p \vee ((\neg p) \wedge q)) \leftrightarrow (p \vee q)$	$(p \wedge ((\neg p) \vee q)) \leftrightarrow (p \wedge q)$
$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$	$(p \rightarrow (q \rightarrow r)) \leftrightarrow ((p \wedge q) \rightarrow r)$
$(p \rightarrow q) \vee (q \rightarrow r)$	

Definitions A formula φ is *satisfiable* iff $\nu(\varphi) = 1$ for some valuation ν. A formula is *unsatisfiable* iff it is not satisfiable, i.e. $\nu(\varphi) = 1$ for *no* valuation ν, or equivalently $\nu(\varphi) = 0$ for every valuation ν.

Observation A formula φ is a tautology iff $\neg\varphi$ is unsatisfiable.

Although it is not hard to prove this observation, we offer some pictures as mental models. Evidently, the truth table method can be adapted to determine whether or not a formula is satisfiable.

tautology

valuations	**truth values**
00…0	**1**
00…1	**1**
.	.
.	.
.	.
11…0	**1**
11…1	**1**

satisfiable

valuations	**truth values**
00…0	
00…1	
.	.
.	**1**
.	.
11…0	
11…1	

unsatisfiable

valuations	**truth values**
00…0	**0**
00…1	**0**
.	.
.	.
.	.
11…0	**0**
11…1	**0**

Refutation Tree Method

This method, which is also called the *semantic tableaux* method, provides a convenient way to determine whether a formula is a tautology, by determining whether the negation of the formula is satisfiable. It does not require constructing a truth table. Later we will also extend it to the predicate calculus.

The method involves constructing a tree with the *negation* of the formula to be checked for being a tautology as the root. The tree is expanded from the root downward. Each node in the tree is replaced at most once by other simpler formulas, and when it is replaced, the node is *checked off* or *numbered* to so indicate.

The rules for replacement are as follows:

¬¬Rule	Replace ¬¬φ with φ on all paths below.
∧Rule	Replace (φ ∧ ψ) with two separate nodes φ and ψ stacked on all paths below.
¬∨Rule	Replace ¬(φ ∨ ψ) with two separate nodes ¬φ and ¬ψ stacked on all paths below.
¬→Rule	Replace ¬(φ → ψ) with two separate nodes φ and ¬ψ stacked on all paths below.
∨Rule	Split (φ ∨ ψ) into two separate paths, φ and ψ, for each path below.
¬∧Rule	Split ¬(φ ∧ ψ) into two separate paths, ¬φ and ¬ψ, for each path below.
→Rule	Split (φ → ψ) into two separate paths, ¬φ and ψ, for each path below.
↔Rule	Split (φ ↔ ψ) into two separate paths, with φ and ψ stacked on one path and ¬φ and ¬ψ stacked on the other, for each path below.
¬↔Rule	Split ¬(φ ↔ ψ) into two separate paths, with ¬φ and ψ stacked on one path and φ and ¬ψ stacked on the other, for each path below.

Atomic formulas and their negations cannot be replaced or split.

A path from the root is *closed* if it contains a formula and its negation. Closed paths are marked by placing ⊥ at the end of the path. Once a path becomes closed, it is not further extended by replacements or splits.

Main Result: The formula at the root is unsatisfiable iff a state of expansion is reached wherein all paths are closed.

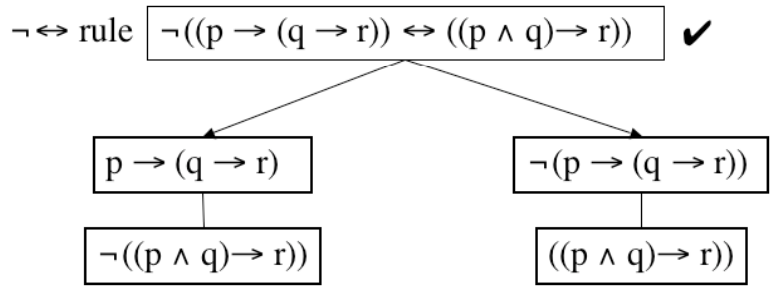
Example: Determine whether or not (p→ (q → r)) ↔ ((p ∧ q) → r) is a tautology using the refutation tree method.

We start the tree with the *negation* of the formula of interest:

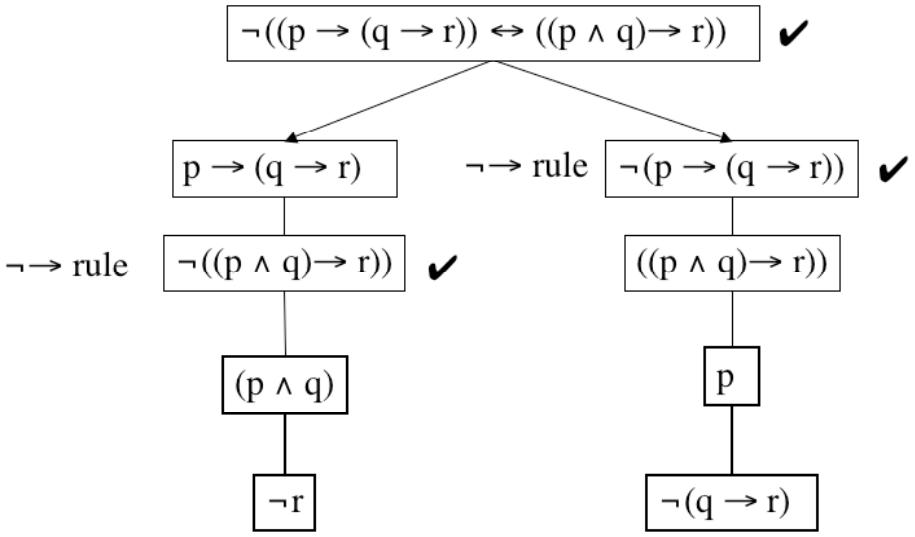
¬((p → (q → r)) ↔ ((p ∧ q)→ r))

For sake of compactness, we'll construct the tree in a breadth-first manner, although it would be more usual to do it depth-first.

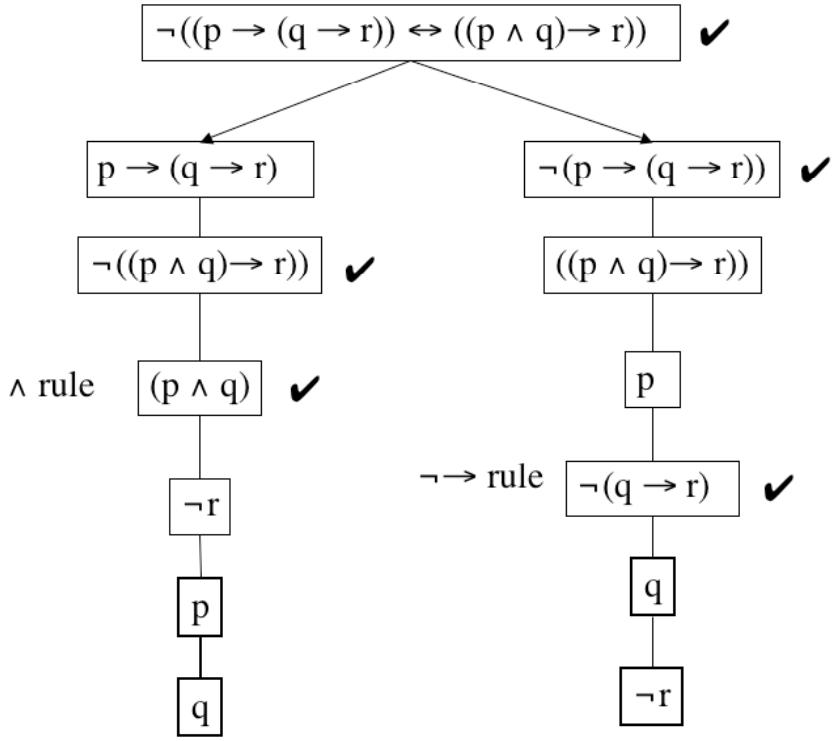
Use the ¬↔rule to split and stack, creating two paths below the root, which now gets checked:



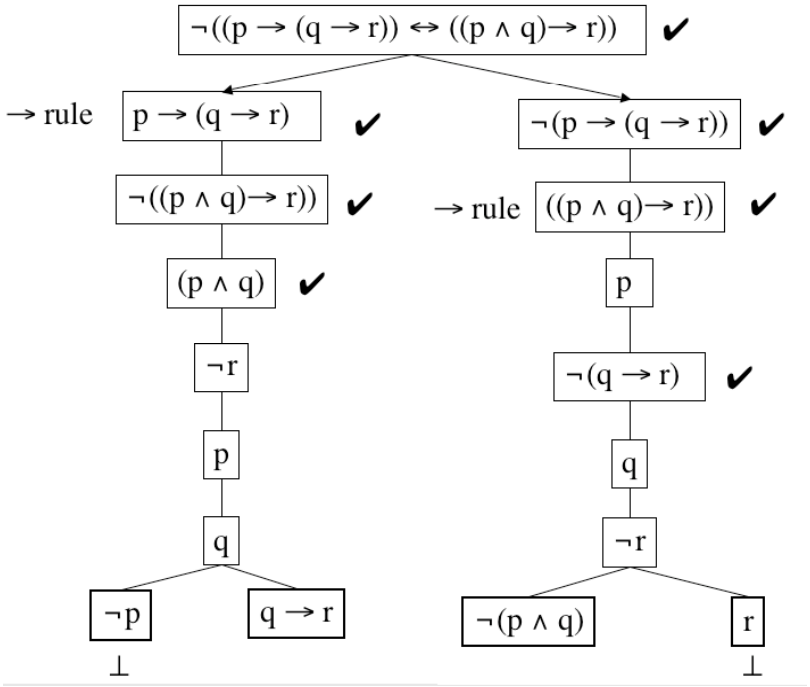
Given a choice, it is better to use a stacking rule over a splitting rule, to prevent the tree from growing wider than necessary. On the left, use the ¬→rule on ¬(p→ (q→ r)) to stack. On the right, the same rule on ¬((p ∧ q)→ r):



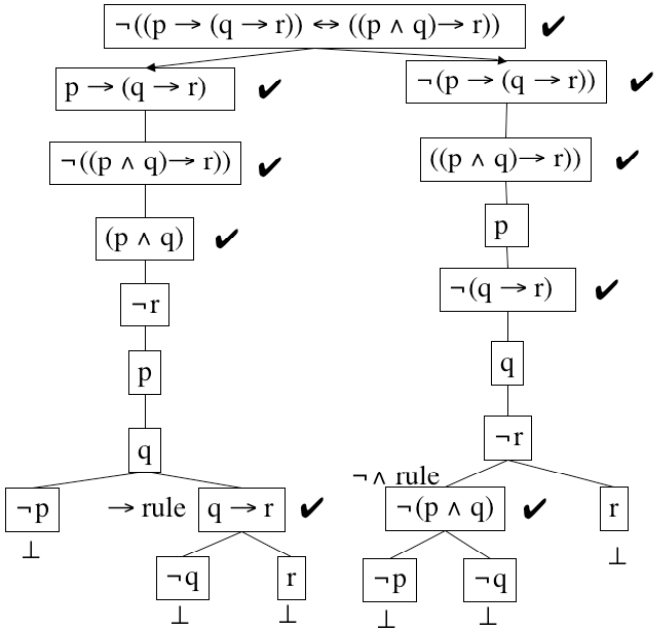
On the left use the ∧ rule, and on the right the ¬→rule:



On the left, use the → rule g on (p→ (q→ r)), and on the right use it on ((p∧ q)→ r), splitting in both cases:



Two paths now become closed, the first due to the presence of p and ¬p and the second due to r and ¬r. Apply the appropriate rules to the remaining unchecked, non-atomic nodes.



Now the remaining paths close, due to q and ¬p, r and ¬r, and p and ¬p.

Determining Counterexamples

When a tree is expanded as far as possible and some path does not close, the root formula is satisfiable. A valuation that induces the value 1 can be determined by assigning 1 to proposition symbols that appear on the path non-negated and 0 to ones that appear on the path negated. Assignment to other proposition variables is arbitrary. Such a valuation is a counterexample to the negation of the root formula being a tautology.

Example Show that $(p \to q) \to ((\neg p) \to (\neg q))$ is not a tautology, and give a counter example.

We'll test the negation of the formula using the tree method. This time we'll just use a table, rather than show the tree explicitly. First use the $\neg\to$rule:

$\neg((p \to q) \to ((\neg p) \to (\neg q)))$ ✔
$(p \to q)$
$\neg((\neg p) \to (\neg q))$

Next use the $\neg\to$rule again, since stacking is preferred, then use the $\neg\neg$rule:

$\neg((p \to q) \to ((\neg p) \to (\neg q)))$ ✔
$(p \to q)$
$\neg((\neg p) \to (\neg q))$ ✔
$\neg p$ $\neg\ (\neg q)$ ✔
q

Now use the $\to$rule g, splitting $p \to q$:

$\neg((p \to q) \to ((\neg p) \to (\neg q)))$ ✔	
$(p \to q)$ ✔	
$\neg((\neg p) \to (\neg q))$ ✔	
$\neg p$ $\neg\ (\neg q)$ ✔	
q	
$\neg p$	q

Now no further expansion is possible, but no path is closed. We can read the counterexample valuation from either path: $\nu(p) = 0$, since p appears negated, and $\nu(q) = 1$, since q appears non-negated.

We see that this valuation induces, in the original formula $(p \to q) \to ((\neg p) \to (\neg q))$, the value determined by $((0 \to 1) \to ((\neg 0) \to (\neg 1))) = ((0 \to 1) \to (1 \to 0)) = (1 \to 0) = 0$, and thus is a counterexample to the formula being a tautology.

In brief, the reason the refutation tree method works is that a valuation satisfies a node iff it simultaneously satisfies the nodes on all paths below. So the root is unsatisfiable iff there is no path where all nodes are satisfied by the same valuation, which is true iff all paths close. See Appendix B for a summary table, and Appendix A for a proof of the method.

Proof Methods

A *proof* of a formula is a step-by-step derivation of a formula according to certain syntactic rules. As such, proofs do not use truth functions directly, although the latter may be used heuristically to guide the proof.

The main families of proof methods are:

- Classical deduction (or Hilbert-Ackermann system)
- Natural deduction (or Gentzen system)

The families are equivalent, but natural deduction is closer to proofs that might appear in a mathematical paper, so we use it. Also, natural deduction strongly parallels certain other ideas that appear in computer science. Tableau methods, such as in the previous section, can also be regarded as a type of proof method, in some sense "dual" to natural deduction.

In natural deduction, the rules are aligned to the connectives. For each connective, there is an *introduction rule* and an *elimination rule*.

The rules derive a formula from other formulas, and the entire derivation can be expressed either in a *tabular form* or as a *tree*. The tabular form is probably the most natural to read, as it is readable step-by-step although the tree representation sometimes, has an appealing succinctness. Regardless of which representation is used, the *rules* themselves are presented in a manner suggestive of a tree, with the *antecedents* of a rule being given above a line and the *consequent* given below.

The rules for the ∧ connective are:

∧-Introduction:

$$\frac{\varphi \qquad \psi}{\varphi \wedge \psi} \wedge I$$

This rule says that if φ and ψ are derived separately, then may φ ∧ ψ be derived.

∧-Elimination:

$$\frac{\varphi \wedge \psi}{\varphi} \wedge E \qquad\qquad \frac{\varphi \wedge \psi}{\psi} \wedge E$$

This rule has two *variants*, left and right. We use the same name for both variants, expecting the reader to determine which one is being used.

Example: Below is a tabular proof using just the rules introduced so far:

$$((p \wedge q) \wedge r) \vdash (p \wedge (q \wedge r))$$

The reading of |- (*turnstyle*) is "from the formula or formulas on the left of |- (called the *premise* or *premises*), the formula on the right (called the *conclusion*) can be derived." This form of statement is sometimes called a *sequent*. The proof of this sequent in tabular form is:

Step	**Formula**	**Justification**
1	$((p \wedge q) \wedge r)$	Premise
2	$(p \wedge q)$	1, ∧-Elimination
3	r	1, ∧-Elimination
4	p	2, ∧-Elimination
5	q	2, ∧-Elimination
6	$(q \wedge r)$	5, 3, ∧-Introduction
7	$(p \wedge (q \wedge r))$	4, 6, ∧-Introduction

Here is the same proof in tree form, where we use ∧E and ∧I rather than spelling out elimination and introduction.:

$$\dfrac{\dfrac{\dfrac{((p \wedge q) \wedge r)}{(p \wedge q)}\wedge E}{p}\wedge E \qquad \dfrac{\dfrac{\dfrac{((p \wedge q) \wedge r)}{(p \wedge q)}\wedge E}{q}\wedge E \qquad \dfrac{((p \wedge q) \wedge r)}{r}\wedge E}{(q \wedge r)}\wedge I}{p \wedge (q \wedge r)}\wedge I$$

The root of the tree is at the bottom, and the leaves must be premises. With trees, the practice is not to number the steps, except in certain special circumstances, to be described soon.

We proceed with giving rules for other connectives.

→-Elimination:

$$\dfrac{\varphi \qquad \varphi \rightarrow \psi}{\psi}\rightarrow E$$

Note that this rule has two antecedents, similar to what we had before with ∧–Introduction, even though this is an elimination rule rather than an introduction rule.

While we're at it, we mention a special case of →-Elimination where ψ is ⊥. Recall that φ → ⊥ is the same as ¬φ, so we call the special case ¬-Elimination.

¬-Elimination:

$$\frac{\varphi \qquad \neg\varphi}{\bot} \neg E$$

Note that this rule might also have been called "⊥-Introduction", but we'll stick with the first name.

Returning to rules for →, the corresponding rule for introduction is different from anything we've seen so far, in that it uses a *sub-derivation*, which makes a *temporary assumption* that doesn't have to be otherwise justified.

→-Introduction:

$$\frac{\begin{array}{c}[\varphi] \\ \vdots \\ \psi\end{array}}{\varphi \rightarrow \psi} \rightarrow I$$

Here φ . . . ψ represents a sub-derivation that begins with *assumption* (as indicated by the brackets [ ]) φ and concludes with ψ. The derived formula is φ → ψ. Applying this rule *discharges* the assumption, which means that outside the sub-derivation, φ, and anything derived from it, *cannot be used* (unless it happens to be derivable by some other means). Every temporary assumption must be discharged before the proof is complete. However, formulas *outside* the sub-derivation may be used in deriving ψ. Note the similarity to *scope rules* in certain programming languages.

In the tabular representation, sub-derivations are represented by *inner boxes*, or by *indenting* the steps of the sub-derivation relative to the use of φ → ψ.

Example: Below is a tabular proof using just the rules introduced so far:

Prove $((p \wedge q) \rightarrow r) \vdash (p \rightarrow (q \rightarrow r))$

Note that some rules mention the *numbers* of the antecedents, while rules requiring a *sub-derivation* mention the *range* of numbers corresponding to the sub-derivation. An assumption is effectively *discharged* outside the box in which it was first made.

Step	Formula	Justification
1	$((p \wedge q) \rightarrow r)$	Premise
2	p	Assumption
3	q	Assumption
4	$p \wedge q$	2, 3, $\wedge$-Introduction
5	r	1, 4, $\rightarrow$-Elimination
6	$q \rightarrow r$	3-5, $\rightarrow$- Introduction
7	$p \rightarrow (q \rightarrow r)$	2-6, $\rightarrow$- Introduction

Below is the same derivation as a tree. In a tree, assumptions are shown in brackets […] with an attached number. Each assumption must be *discharged* (or *cancelled*) before the proof is complete. When a rule that discharges an assumption is used, the number of the assumption is shown with the rule.

$$\dfrac{\dfrac{\dfrac{[p]_2 \quad [q]_1}{p \wedge q}\wedge I \quad ((p \wedge q) \rightarrow r)}{r}\rightarrow E}{q \rightarrow r}\rightarrow I_1 \qquad \textit{discharges } q$$

$$p \rightarrow (q \rightarrow r) \qquad \rightarrow I_2 \qquad \textit{discharges } p$$

In constructing the tree, we bracketed the assumptions first without numbering them, then added the numbers in the order in which the assumptions are discharged. Notice that the original premise does not get discharged.

$\rightarrow$-Introduction Degenerate Case

If we have *already derived* ψ and wish to use a formula $\varphi \rightarrow \psi$ for some reason, the rule allows introduction of the latter appealing to $\rightarrow$-Introduction *without* a sub-derivation.

$\rightarrow$-Introduction degenerate case:

$$\dfrac{\psi}{\varphi \rightarrow \psi}\rightarrow I \qquad \textit{nothing discharged}$$

The rationale is that ψ, as already derived, could have been the conclusion of a sub-derivation with an empty assumption. In this case, there is no assumption to discharge.

$\neg$-Introduction as a Special Case of $\rightarrow$-Introduction

Recall that $\neg\varphi$ is regarded as an abbreviation for $\varphi \rightarrow \perp$. Thus if $\perp$ appears as the last line of a sub-derivation that begins with assumption φ, we can show the derived formula as $\neg\varphi$.

¬-Introduction:

$$\dfrac{\begin{array}{c}[\varphi]\\ \vdots\\ \bot\end{array}}{\neg\varphi}\ \neg I$$

This is one form of *proof by contradiction.*

RAA Rule

Another form of *proof by contradiction* uses a different rule having a similarity to ¬I: RAA, for *reductio ad absurdum.*

RAA:

$$\dfrac{\begin{array}{c}[\neg\varphi]\\ \vdots\\ \bot\end{array}}{\varphi}\ RAA$$

Example using the RAA rule: |- $(\neg\neg p \rightarrow p)$

Proof:

$$\dfrac{\dfrac{[\neg\neg p]_2 \qquad [\neg p]_1}{\bot}\ \neg E}{p}\ RAA_1$$
$$\neg\neg p \rightarrow p \qquad \rightarrow I_2$$

The reason that RAA and are not the same is because we are dealing with syntactic entities. The ¬–Introduction rule is *not* an instance of RAA, as it does not start with ¬φ and derive φ. We are not allowed to impose our informal knowledge of meaning in formal rules.

The RAA rule is considered controversial by some, because it suggests that something can be proved by assuming its negation, then deriving ⊥. This rule is disallowed in a subset of logic known as *intuitionistic logic*, which insists on constructive proofs. So ¬I is allowed in intuitionistic logic. Rules that are not intuitionistic are sometimes referred to as *classical.*

Rules for ⊥

If ⊥ can be derived, then anything can be derived:

⊥-Elimination:

$$\frac{\bot}{\varphi}\ \bot E$$

This rule is sometimes simply called the ⊥ rule, rather than ⊥-elimination rule. Normally this rule will only get used *inside* sub-derivations, based on an assumption made at the start, since a system in which everything is derivable is not very interesting.

Example: p |- (¬p → q)

Proof as a tree:

$$\dfrac{\dfrac{p \qquad [\neg p]_1}{\bot}\ \neg E}{\dfrac{q}{\neg p \rightarrow q}\ \rightarrow I_1}\ \bot E$$

Proof in tabular form:

Step	Formula	Justification
1	p	Premise
2	¬p	Assumption
3	⊥	1, 2 ¬E
4	q	3, ⊥E
5	¬p → q	2-4, →I

Formulas vs. Proposition Symbols

After doing a few proofs, it becomes clear that a proof of the same derivation pattern could be carried out with *arbitrary* formulas replacing the various proposition symbols. So we will adopt the practice of expressing sequents and their proofs in a more *generic* form, wherein variables representing formulas φ, ψ, … are preferred over proposition symbols p, q, … . Thus the previous sequent would be expressed generically as:

$$\varphi \vdash (\neg\varphi \rightarrow \psi)$$

The proof would have the same form as before, e.g. as a tree:

$$\dfrac{\dfrac{\varphi \qquad [\neg\varphi]_1}{\bot}\,\neg E}{\psi}\,\bot E$$
$$\neg\varphi \rightarrow \psi \qquad \rightarrow I_1$$

Sequents with an Empty Antecedent

If the antecedent is empty, that means that the consequent is derivable without any premises.

Examples: |- $(\varphi \rightarrow \neg\neg\varphi)$
|- $(\neg\neg\varphi \rightarrow \varphi)$

Proof of the first sequent:

$$\dfrac{\dfrac{[\varphi]_2 \qquad [\neg\varphi]_1}{\bot}\,\neg E}{\neg\neg\varphi}\,\neg I_1$$
$$\varphi \rightarrow \neg\neg\varphi \qquad \rightarrow I_2$$

Note that we chose to use ¬I in the second step so as to discharge the assumption $[\neg\varphi]_1$. Had we chosen ⊥E instead, the assumption would not get discharged and the proof would not be complete.

Derived Rules

Obviously it is quite possible that a given derivation pattern can recur with different formulas in a large proof. Rather than re-instantiate that derivation, it is often more succinct to think of the pattern as a rule in its own right, i.e. a *derived rule*. Obviously we can use any sequent as if it were such a rule. For example, if we have derived a sequent such as

φ |- $(\neg\neg\varphi)$

we might want to use it as a rule, in the form:

$$\dfrac{\varphi}{\neg\neg\varphi}$$

The question is whether we have to invent a *name* for that rule. Sometimes we will want to do so, but to avoid the necessity, we will permit the sequent itself to be used as the name. Of course, the sequent itself must be derivable, but that derivation can be done separately.

Derived rule:

$$\frac{\varphi}{\neg\neg\varphi} \qquad \varphi \vdash (\neg\neg\varphi)$$

The *name* of this derived rule is given by the sequent $\varphi \vdash (\neg\neg\varphi)$ on the right. Of course, we might have chosen a more descriptive name such as "$\neg\neg$-Introduction".

Informally, the sequent is serving as a *lemma* in an overall proof.

Similarly we can also derive the converse of the preceding rule using RAA:

$$\frac{\neg\neg\varphi}{\varphi} \qquad (\neg\neg\varphi) \vdash \varphi$$

Rules for $\vee$

Recall that $\varphi \vee \psi$ abbreviates $\neg(\neg\varphi \wedge \neg\psi)$. Hence we would expect that various rules for $\vee$ could be derived from other rules. Nonetheless, we would like to have rules for introduction and elimination just as for the other connectives.

For introduction, there are two variants, left and right.

$\vee$-Introduction:

$$\frac{\varphi}{\varphi \vee \psi}\ \vee I \qquad\qquad \frac{\psi}{\varphi \vee \psi}\ \vee I$$

Note that both variants *lose information* from the antecedent. Consequently, *these rules will seldom constitute the last rule of a proof.*

The elimination rule for $\vee$ is the most complicated so far, as it involves *two* sub-derivations.

$\vee$-Elimination:

$$\begin{array}{ccc} & [\varphi] & [\psi] \\ & \vdots & \vdots \\ \varphi \vee \psi & \xi & \xi \\ \hline \multicolumn{3}{c}{\xi} \end{array} \quad \vee E$$

Note that the consequent is a formula ξ. To eliminate the formula containing $\vee$, namely $\varphi \vee \psi$, we need two sub-derivations, one that derives ξ from assumption φ and one that derives ξ from assumption ψ. Both assumptions are discharged in the application of this rule. Hence this rule captures the informal notion of *proof by cases*. There is also a similarity to the use of *if … then … else …* in programming.

Here is an example that uses both the introduction and elimination rules for ∨.

Example: (φ ∨ ψ) |- (ψ ∨ φ)

Proof as a tree:

$$\dfrac{(\varphi \vee \psi) \qquad \dfrac{[\varphi]_1}{\psi \vee \varphi}\vee I \qquad \dfrac{[\psi]_2}{\psi \vee \varphi}\vee I}{\psi \vee \varphi}\vee E_{1,2}$$

Note here that ξ in the ∨E rule is identified with (ψ ∨ φ).

Derivation of the ∨E Rule

When we expand the abbreviations for ∨ in the ∨E rule, we have

$$\dfrac{\neg(\neg\varphi \wedge \neg\psi) \qquad \begin{matrix}[\varphi] \\ \vdots \\ \xi\end{matrix} \qquad \begin{matrix}[\psi] \\ \vdots \\ \xi\end{matrix}}{\xi}$$

The proof can be re-cast as follows:

$$\dfrac{\neg(\neg\varphi \wedge \neg\psi) \qquad \dfrac{\dfrac{\begin{matrix}[\varphi]_1 \\ \vdots \\ \xi\end{matrix} \quad [\neg\xi]_3}{\dfrac{\perp}{\neg\varphi}\neg E}\neg I_1 \qquad \dfrac{\dfrac{\begin{matrix}[\psi]_2 \\ \vdots \\ \xi\end{matrix} \quad [\neg\xi]_3}{\dfrac{\perp}{\neg\psi}\neg E}\neg I_2}{\neg\varphi \wedge \neg\psi}\wedge I}{\perp}\neg E}{\xi}RAA_3$$

Note that the discharge of assumptions φ is deferred until after ⊥ is introduced, to enable ¬φ and ¬ψ to be derived by ¬I.

Below is an equivalent derivation in tabular form.

1	$\neg(\neg\varphi \wedge \neg\psi)$	Premise
2	$\neg\xi$	Assumption
3	φ	
4	. . .	Assumed sub-derivation
5	ξ	
6	$\bot$	2, 5, $\neg$E
7	$\neg\varphi$	3-6, $\neg$I
8	ψ	
9	. . .	Assumed sub-derivation
10	ξ	
11	$\bot$	2, 10, $\neg$E
12	$\neg\psi$	8-11, $\neg$I
13	$(\neg\varphi \wedge \neg\psi)$	7, 12, $\wedge$I
14	$\bot$	13, 1, $\neg$E
15	ξ	2-14, RAA

Rules for $\leftrightarrow$

Since $\leftrightarrow$ is an abbreviation expressed in terms of $\rightarrow$, we expect the rules $\leftrightarrow$ for to be similar to those for $\rightarrow$, but more symmetric.

$\leftrightarrow$-Introduction:

$$\begin{array}{cc} [\varphi] & [\psi] \\ \vdots & \vdots \\ \psi & \varphi \\ \hline \multicolumn{2}{c}{\varphi \leftrightarrow \psi} \end{array} \quad \leftrightarrow\text{I}$$

$\leftrightarrow$-Elimination:

This rule has two variants:

$$\frac{\varphi \quad \varphi \leftrightarrow \psi}{\psi} \leftrightarrow\text{E} \qquad \frac{\psi \quad \varphi \leftrightarrow \psi}{\varphi} \leftrightarrow\text{E}$$

These rules may be derived by expanding the abbreviation $\leftrightarrow$.

Other Useful Derivations

Law of the Excluded Middle (LEM)

This sequent requires RAA for its derivation. It can also be used as a rule.

$\vdash (\varphi \vee \neg\varphi)$ LEM

Although it is tempting to try ∨I as the last rule in the derivation, it was mentioned that this seldom works, because ∨I loses information. Instead we plan to use RAA, in the form:

$$\begin{array}{l} [\neg(\varphi \vee \neg\varphi)] \\ \quad . \\ \quad . \\ \quad . \\ \quad \bot \\ \hline (\varphi \vee \neg\varphi) \end{array} \quad \text{RAA}$$

$$\dfrac{\dfrac{\dfrac{[\neg(\varphi \vee \neg\varphi)]_2 \quad \dfrac{[\varphi]_1}{\varphi \vee \neg\varphi}\vee\text{I}}{\bot}\neg\text{E}}{\neg\varphi}\neg\text{I}_1 \quad [\neg(\varphi \vee \neg\varphi)]_2}{\dfrac{\varphi \vee \neg\varphi}{\bot}\neg\text{E}}\vee\text{I}$$

$$(\varphi \vee \neg\varphi) \qquad \text{RAA}_2$$

Note that in this derivation, the assumption $\neg(\varphi \vee \neg\varphi)$ is used twice.
Here is the same derivation in tabular form:

1		$\neg(\varphi \vee \neg\varphi)$	Assumption
2		φ	Assumption
3		$\varphi \vee \neg\varphi$	2, ∨I
4		$\bot$	3, 1, ¬E
5		$\neg\varphi$	2-4, ¬I
6		$\varphi \vee \neg\varphi$	5, ∨I
7		$\bot$	6, 1, ¬E
8	$\varphi \vee \neg\varphi$		1-7, RAA

DeMorgan's Laws

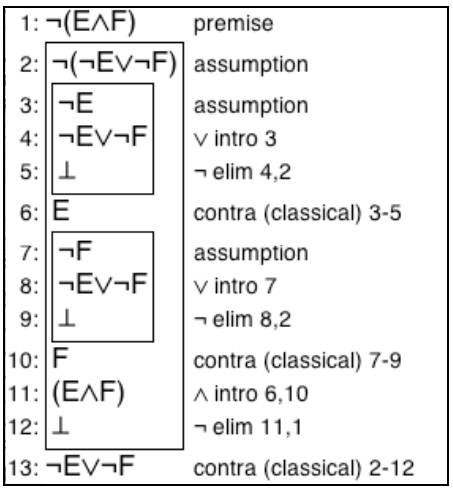
There are several variants on DeMorgan's Laws:

a. $\neg(\varphi \wedge \psi) \vdash (\neg\varphi) \vee (\neg\psi)$	b. $(\varphi \wedge \psi) \vdash \neg((\neg\varphi) \vee (\neg\psi))$
c. $(\neg\varphi) \vee (\neg\psi) \vdash \neg(\varphi \wedge \psi)$	d. $\neg((\neg\varphi) \vee (\neg\psi)) \vdash (\varphi \wedge \psi)$
e. $\neg(\varphi \vee \psi) \vdash (\neg\varphi) \wedge (\neg\psi)$	f. $(\varphi \vee \psi) \vdash \neg((\neg\varphi) \wedge (\neg\psi))$
g. $(\neg\varphi) \wedge (\neg\psi) \vdash \neg(\varphi \vee \psi)$	h. $\neg((\neg\varphi) \wedge (\neg\psi)) \vdash (\varphi \vee \psi)$

Proof of variant a:

1	$\neg(\varphi \wedge \psi)$	Premise
2	$\quad\neg((\neg\varphi) \vee (\neg\psi))$	Assumption
3	$\quad\quad\neg\varphi$	Assumption
4	$\quad\quad(\neg\varphi) \vee (\neg\psi)$	3, ∨I
5	$\quad\quad\bot$	4, 2, ¬E
6	$\quad\varphi$	3-5, RAA
7	$\quad\quad\neg\psi$	Assumption
8	$\quad\quad(\neg\varphi) \vee (\neg\psi)$	7, ∨I
9	$\quad\quad\bot$	8, 2, ¬E
10	$\quad\psi$	7-9, RAA
11	$\quad\varphi \wedge \psi$	6, 10, ∧I
12	$\quad\bot$	11, 1, ¬E
13	$(\neg\varphi) \vee (\neg\psi)$	2-12, RAA

Below is essentially the same proof constructed using the JAPE Proof Editor. (Some of the rules have slightly different names than what we are using. For example, RAA is called "contra (classical)".



Proof of variant c:

1	$(\neg\varphi) \vee (\neg\psi)$	Premise
2	$(\varphi \wedge \psi)$	Assumption
3	φ	2, $\wedge$E
4	$\neg\varphi$	Assumption
5	$\bot$	3, 4, $\neg$E
6	ψ	2, $\wedge$E
7	$\neg\psi$	Assumption
8	$\bot$	6, 7, $\neg$E
9	$\bot$	1, 4-5, 7-8, $\vee$E
10	$\neg(\varphi \wedge \psi)$	2-9, $\neg$I

It is interesting that variant *a* requires RAA, but variant *c*, the converse of *c*, does not. Below is a similar derivation constructed using the JAPE proof editor.

1:	$(\neg E)\vee(\neg F)$	premise
2:	$E\wedge F$	assumption
3:	$\neg E$	assumption
4:	E	$\wedge$ elim 2
5:	$\bot$	$\neg$ elim 4,3
6:	$\neg F$	assumption
7:	F	$\wedge$ elim 2
8:	$\bot$	$\neg$ elim 7,6
9:	$\bot$	$\vee$ elim 1,3-5,6-8
10:	$\neg(E\wedge F)$	$\neg$ intro 2-9

It is suggested that you try to prove the other variants. Variant *f* requires no proof.

Appendix A: Proof of Correctness of the Refutation Tree Method

This proof presupposes familiarity with the method, which is not stated here. It is clear that it is always possible to complete a tree using the rules, as the root formula is finite and applying a rule only extends the tree by adding shorter formulas. We thus need to prove:

I. If φ is satisfiable and is at the root of a complete tree, then there is an open path in the tree.

II. If φ is the root of a complete tree with an open path, then φ is satisfiable.

Proof of I: Suppose that φ at the root of a complete tree is satisfiable. Let υ be a valuation that satisfies φ. We show that at every step of construction, there is a path from the root on which every formula is satisfied by υ. This is true initially, since φ is the only formula on a path then. We can verify that the application of any rule extends at least one path in the tree by adding only formula(s) to the end that are satisfied by υ. For example, in the rule for $\varphi \vee \psi$, if υ satisfies $\varphi \vee \psi$ and all its predecessors in the path, then υ must satisfy at least one of φ or ψ as well. Similarly, in the rule for $\varphi \wedge \psi$, if υ satisfies $\varphi \wedge \psi$ along with all predecessors in the path, then it must satisfy both φ and ψ. Thus by induction on the number of steps in tree formation, there is always a path from the root to a leaf on which all formulas are satisfied by υ. But a closed path cannot have this property, since by definition of "closed" it must contain $\perp$ derived from a formula σ along with the negation of that formula $\neg\sigma$. Hence, based on the supposition that the root is satisfiable, even in a complete tree, there is at least one open path.

Proof of II: Suppose that there is a complete tree with φ as root having an open path Γ. Since Γ is open, for any atom σ, at most one of σ or $\neg\sigma$ occurs on Γ. Let υ be any valuation such that $\upsilon(\sigma) = 1$ if σ occurs on Γ, and $\upsilon(\sigma) = 0$ if $\neg\sigma$ occurs on Γ. We claim that *every* formula, including the root, on Γ is satisfied by υ. We work our way up Γ from its leaf by induction, establishing that each formula is satisfied by υ. By definition, the leaf is satisfied by υ, as it must be an atom or the negation of an atom. (A non-atom can end a path only if its negation also appears on the path, but then the path could not be open.) Now if ρ is any formula on Γ, all of whose successors on the path are satisfied by υ, then ρ itself must be satisfied by υ, by considering each of the rule types. For example, if ρ were of the form $\varphi \vee \psi$, it would be followed on Γ by either φ or ψ, which is satisfied according to the induction hypothesis. Thus ρ is satisfied by υ. Similarly, if ρ were of the form $\varphi \wedge \psi$, then both φ and ψ would follow on Γ and are satisfied by υ according to the induction hypothesis, so ρ itself is satisfied by υ. A similar analysis holds for the other rule types. Thus by induction, all nodes in Γ, including the root of the tree, are satisfied by υ.

Appendix B: Satisfiability Checking Rules

Refutation Tree or Analytic Tableau Method

φ is satisfiable iff there is a complete tree with φ at the root and an open path.

(φ is a tautology iff $\neg\varphi$ is *not* satisfiable.)

$\vee$	$\varphi \vee \psi$ / \\ φ ψ	$\neg(\varphi \vee \psi)$ | $\neg\varphi$ | $\neg\psi$
$\rightarrow$	$\varphi \rightarrow \psi$ / \\ $\neg\varphi$ ψ	$\neg(\varphi \rightarrow \psi)$ | φ | $\neg\psi$
$\wedge$	$\neg(\varphi \wedge \psi)$ / \\ $\neg\varphi$ $\neg\psi$	$\varphi \wedge \psi$ | φ | ψ
$\leftrightarrow$	$\varphi \leftrightarrow \psi$ / \\ φ $\neg\varphi$ | | ψ $\neg\psi$	$\neg(\varphi \leftrightarrow \psi)$ / \\ φ $\neg\varphi$ | | $\neg\psi$ ψ
$\neg$	φ . . . $\neg\varphi$ $\perp$ (closed)	$\neg\neg\varphi$ | φ

Each rule has the property that a valuation that satisfies the parent also satisfies the successors on at least one of the paths.

Appendix C: Natural Deduction Rules

Connective	Introduction	Elimination
$\wedge$	$\dfrac{\varphi \quad \psi}{\varphi \wedge \psi}\ (\wedge I)$	$\dfrac{\varphi \wedge \psi}{\psi}\ (\wedge E) \qquad \dfrac{\varphi \wedge \psi}{\varphi}\ (\wedge E)$
$\vee$	$\dfrac{\varphi}{\varphi \vee \psi}\ (\vee I) \qquad \dfrac{\psi}{\varphi \vee \psi}\ (\vee I)$	$\dfrac{\varphi \vee \psi \quad [\varphi] \ldots \xi \quad [\psi] \ldots \xi}{\xi}\ (\vee E)$
$\rightarrow$	$\dfrac{[\varphi] \ldots \psi}{\varphi \rightarrow \psi}\ (\rightarrow I)$	$\dfrac{\varphi \quad \varphi \rightarrow \psi}{\psi}\ (\rightarrow E)$
$\leftrightarrow$	$\dfrac{[\varphi] \ldots \psi \quad [\psi] \ldots \varphi}{\varphi \leftrightarrow \psi}\ (\leftrightarrow I)$	$\dfrac{\varphi \quad \varphi \leftrightarrow \psi}{\psi}\ (\leftrightarrow E) \qquad \dfrac{\psi \quad \varphi \leftrightarrow \psi}{\varphi}\ (\leftrightarrow E)$
$\neg$	$\dfrac{[\varphi] \ldots \bot}{\neg\varphi}\ (\neg I)$	$\dfrac{\varphi \quad \neg\varphi}{\bot}\ (\neg E)$
$\bot$		$\dfrac{\bot}{\varphi}\ (\bot E)$
		$\dfrac{[\neg\varphi] \ldots \bot}{\varphi}\ (RAA)$