Appendix to “A Primer on *Rational Consequence Relations*, *Popper Functions*, and their Ranked Structures”

James Hawthorne
University of Oklahoma

Abstract of the paper. *Rational consequence relations* and *Popper functions* provide logics for reasoning under uncertainty, the former purely qualitative, the latter probabilistic. But few researchers seem to be aware of the close connection between these two logics. I'll show that *Popper functions* are probabilistic versions of *rational consequence relations*. I'll not assume that the reader is familiar with either logic. I present them, and explicate the relationship between them, from the ground up. I'll also present alternative axiomatizations for each logic, showing them to depend on weaker axioms than usually recognized.

This appendix contains formal statements of definitions and results cited in the paper.

1. Section 3

Definition 1: *Ranked Truth-Table*: A ranked truth-table is the usual kind of truth-table for a finite language for sentential logic supplemented with an additional column, the *rank* column, which indicates the rank of each truth-table line (i.e. each truth-value assignment). The highest rank is rank-1; one or more lines are marked rank-1, indicated by placing a ‘1’ beside them in the rank column. If any lines remain unranked, some (or all) of them may be marked rank-2 by placing a ‘2’ beside them in the rank column. If any lines remain unranked, some (or all) of them may be marked as rank-3 by placing a ‘3’ beside them in the rank column. ... This sequential, numerical ranking may stop with rank-1, or may stop at any point before all lines have been numerically ranked, or may continue until all lines have been numerically ranked. If some lines remain numerically unranked (i.e., not marked by a finite positive integer), they are assigned rank-ω, and the symbol 'ω' is placed beside them in the column.

Definition 2: *Truth-table Consequence Relation*: The *truth-table consequence relation* generated by ranked truth-table *T* is the relation $|\!\sim_T$ such that, for all sentences A and B in *T*’s language,

B $|\!\sim_T$ A *if and only if*

1. the rank of *B* is ω; or
2. the rank of *B* is not ω, and every rank-B line that makes B true also makes A true.

Definition 3: *Probabilistic Ranked Truth-Table*: A probabilistic ranked truth-table is a ranked truth-table (as defined above) for a finite language for sentential logic supplemented with an additional column, the *weight* column, which indicates the weight of each truth-table line (i.e. each truth-value assignment). Rank-ω lines remain unweighted – the weight column beside each rank-ω lines is left empty. The weight column beside each line having a finite rank contains some positive real number.

Definition 4: *Truth-table Probability Function*: The truth-table conditional probability function generated by probabilistic ranked truth-table T_P is the function P such that for all sentences A and B in T_P’s language,

P[A | B] = r if and only if

1. the rank of B is ω and r = 1; or
2. r = the sum of the weights of the rank-B lines that make (A·B) true divided by
   the sum of the weights of the rank-B lines that make B true.

2. Section 4

Definition 5: *Usual Axioms for the Preferential Consequence Relations*: Let *L* be a language having the syntax of sentential logic. A *Preferential Consequence Relation* on *L* is any relation between pairs of sentences of *L* that satisfies the following axioms:

0. for some E, F, E |/~ F	NT	(Non-Triviality)
1. A |~ A	R	(Reflexivity)
2. If B |= C, C |= B, B |~ A, then C |~ A	LCE	(Left Classical Equivalence)
3. If C |~ B, B |= A, then C |~A	RW	(Right Weakening)
4. If C |~ A, B |~ A, then (C∨B) |~ A	OR	(left disjunction)
5. If B |~ A, B |~ C, then (B·C) |~ A	CM	(Cautious Monotonicity)
6. If C |~ B, C |~ A, then C |~ (B·A)	AND	(right conjunction)

Definition 6. *Usual Axioms for the Rational Consequence Relations*: Let *L* be a language having the syntax of sentential logic. A *Rational Consequence Relation* on *L* is any relation between pairs of sentences of *L* that satisfies the axioms (0-6 above) for the *preferential consequence relations* together with the following axiom:

7. If B |~ A, B |/~ ¬C, then (B·C) |~ A	RM	(Rational Monotony)

Definition 7: *Weak Axioms* for the *Rational Consequence Relations*: Let *L* be a language having the syntax of sentential logic. A *Weak Rational Consequence Relation* on *L* is any relation between pairs of sentences of *L* that satisfies the following axioms:

0. for some E, F, *not* E |~ F	NT	
1. A |~ A	R	
2. If B |= C, C |= B, B |~ A, then C |~ A	LCE	
3. If C |~ B, B |= A, then C |~A	RW	
4. If (B·C) |~ A, (B·¬C) |~ A, then B |~ A	WOR	(Weak OR)
5. If B |~ (C·A), then (B·C) |~ A	VCM	(Very Cautious Monotony)
6.1 If (C·B) |~ A, then (C·B) |~ (B·A)	VWAND	(Very Weak AND)
6.2 If B |~ A, B |~ ¬A, then B |~ C	CNTRA	(ConTRAdiction)
7. If B |~ A, B |/~ ¬C, then (B·C) |~ A	RM	

Definition 8: *Autonomous Axioms* for the *Rational Consequence Relations*:[1] Let *L* be a language having the syntax of sentential logic. An *Autonomous Rational Consequence Relation* on *L* is any relation between pairs of sentences of *L* that satisfies the following axioms:

0. for some E, F, *not* E |~ F	NT	
1. A |~ A	R	
2. If (C·B) |~ A, then (B·C) |~ A	LC	(Left Commutivity)
3.1 If C |~ (B·A), then C |~ B	SMP-L	(Simplification-Left)
3.2 If C |~ (B·A), then C |~ A	SMP-R	(Simplification-Right)
3.3 If B |~ ¬¬A, then B |~ A	DN	(Double Negation)
3.4 If C |~ (¬(B·A) · B), then C |~ ¬A	SYL	(Syllogism)
4. If (B·C) |~ A, (B·¬C) |~ A, then B |~ A	WOR	
5. If B |~ (C·A), then (B·C) |~ A	VCM	
6.1 If (C·B) |~ A, then (C·B) |~ (B·A)	VWAND	
6.2 If B |~ A, B |~ ¬A, then B |~ C	CNTRA	
7. If B |~ A, B |/~ ¬C, then (B·C) |~ A	RM	

Theorem 1: The following rules follow from the Autonomous Axioms for *rational consequence relations*:

(AND): If C |~ B, C |~ A, then C |~ (B·A).
(CM) : If B |~ A, B |~ C, then (B·C) |~ A (Cautious Monotonicity).
(DN+): If B |~ A, then B |~ ¬¬A (Double Negation Addition).
(RC): If C |~ (B·A), then C |~ (A·B) (Right Commutivity).
(RNC): If C |~ ¬(B·A), then C |~ ¬(A·B) (Right Negation Commutivity).
(ADD): If C |~ ¬B, then C |~ ¬(A·B) (Addition).
(LRE): If C |~ B, B |~ C, and B |~ A, then C |~ A (Left Rational Equivalence).

---

[1] These axioms rely solely on *negation* and *conjunction*. Other logical terms for sentential logic (*or*, *if…then*, *if and only if*) may be treated as *defined terms* in the usual way: '(A∨B)' abbreviates '¬(¬A·¬B)', '(A⊃B)' abbreviates '¬(A·¬B)', '(A≡B)' abbreviates '(¬(A·¬B)·¬(¬A·B))'.

(LA): If (C·B)·D |~ A then C·(B·D) |~ A (Left Associativity).

(LR): If (C·B)·D |~ A then (C·D)·B |~ A (Left Reordering).

proof:

(AND): Suppose C |~ B, C |~ A, and C |/~ (B·A) (for reductio).
C |/~ ¬¬(B·A) (3.3, DN), (C·¬(B·A)) |~ B (7, RM), (C·¬(B·A)) |~ (¬(B·A)·B) (6.1, VWAND), (C·¬(B·A)) |~ ¬A (3.4, SYL), (C·¬(B·A)) |~ A (7, RM), (C·¬(B·A)) |~ (B·A) (6.2, CNTRA), (C·(B·A)) |~ (C·(B·A)) (1, R), (C·(B·A)) |~ (B·A) (3.2, SMP), C |~ (B·A) (4, VWOR), contradiction.

(CM): Suppose B |~ A, B |~ C. Then B |~ C·A (AND), so B·C |~ A (VCM).

(DN+): Suppose B |~ A and B |/~ ¬¬A. Then B·¬A |~ A (RM), and B·¬A |~ ¬A (R, SMP), so B·¬A |~ ¬¬A (CNTRA), and B·¬¬A |~ ¬¬A (R, SMP), so B |~ ¬¬A contradiction.

(RC): Suppose C |~ (B·A). Then C |~ B (SMP), C |~ A (SMP), C |~ (A·B) (AND).

(RNC): Suppose C |~ ¬(B·A), C |/~ ¬(A·B). Then C·(A·B) |~ ¬(B·A) (RM), C·(A·B) |~ B (R, SMP, SMP), C·(A·B) |~ ¬(B·A)·B (AND), C·(A·B) |~ ¬A (SYL), C·(A·B) |~ A (R, SMP, SMP), C·(A·B) |~ ¬(A·B) (CNTRA), C·¬(A·B) |~ ¬(A·B) (R, SMP), C |~ ¬(A·B) (WOR) contradiction.

(ADD): Suppose C |~ ¬B, C |/~ ¬(A·B). Then C·(A·B) |~ ¬B (RM), C·(A·B) |~ B (R, SMP, SMP), C·(A·B) |~ ¬(A·B) (CNTRA), C·¬(A·B) |~ ¬(A·B) (R, SMP), C |~ ¬(A·B) (WOR) contradiction.

(LRE) Suppose C |~ B, B |~ C, B |~ A. Then C·B |~ A (CM), C·B |~ ¬¬A (DN+), C·B |~ ¬(¬A·B) (ADD), C·B |~ ¬(B·¬A) (NC); C·¬B |~ ¬B (R, SMP), C·¬B |~ ¬(B·¬A) (ADD); C |~ ¬(B·¬A) (WOR), C |~ ¬(B·¬A)·B (AND), C |~ ¬¬A (SYL), C |~ A (DN).

(LA) Suppose (C·B)·D |~ A. (C·B)·D |~ (C·B)·D (R), so (C·B)·D |~ D (SMP) and (C·B)·D |~ (C·B) (SMP), so (C·B)·D |~ B (SMP) and C·B)·D |~ C (SMP); thus, so far we have (C·B)·D |~ D, (C·B)·D |~ B, (C·B)·D |~ C. Then (C·B)·D |~ C·(B·D) (AND twice). Similarly, C·(B·D) |~ (C·B)·D. Thus C·(B·D) |~ A (LRE).

(LR) Suppose (C·B)·D |~ A. (C·B)·D |~ (C·B)·D (R), so (C·B)·D |~ D (SMP), (C·B)·D |~ (C·B) (SMP), (C·B)·D |~ B (SMP), (C·B)·D |~ C (SMP); thus, so far we have (C·B)·D |~ D, (C·B)·D |~ B, (C·B)·D |~ C. Then (C·B)·D |~ (C·D)·B (AND twice). Similarly, (C·D)·B |~ (C·B)·D. Thus (C·D)·B |~ A (LRE).

Theorem 2: The Autonomous Axioms follow from the Weak Axioms.

proof: The axioms are the same except for axioms 2 and 3. Autonomous Axiom 2 follows easily from Weak Axiom 2. Autonomous axioms 3.1-3.4 follow easily from Weak Axiom 3.

Theorem 3: AND, CM, and OR follow from the Weak Axioms.

proof: AND and CM follow from the Autonomous Axioms (Tm1), which follow from the Weak Axioms (Tm2). So AND and CM follow from the Weak Axioms. We derive OR using AND and the Weak Axioms, as follows:

(OR): If C |~ A, B |~ A, then (C∨B) |~ A.

Suppose C |~ A, B |~ A.

Suppose C |/~ B. Then C |/~ ¬¬B (RW), so C·¬B |~ A (RM), (C∨B)·¬B |~ A (LCE), (C∨B)·B |~ A (LLE from B |~ A), (C∨B) |~ A.

Suppose C |~ B. Then (C∨B)·C |~ B (LLE), (C∨B)·¬C |~ B (R, RW), (C∨B) |~ B (WOR), B |~ (C∨B) (R, RW), (C∨B) |~ A (LRE).

Theorem 4: The Weak Axioms are inter-derivable with the Usual Axioms. The *Weak rational consequence relations* are just the *Usual rational consequence relations*.

proof: The axioms are the same except for 4, 5, and 6. It's easy to derive Weak Axioms 4 (WOR), 5 (VCM), 6.1 (VWAND) and 6.2 (CNTRA) from the Usual Axioms. Theorem 3 shows that Usual Axioms 4 (OR), 5 (CM), and 6 (AND) are derivable from the Weak Axioms.

Definition 9: rc-logical entailment, B |≈ A:

'B |≈ A' abbreviates "for every relation |~ that satisfies the Autonomous Axioms 0-7, B |~ A". Read 'B |≈ A' as "B rc-entails A" – i.e. "B rational-consequence-entails A".

Theorem 5: B |≈ A if and only if B |= A (i.e. B logically entails C).

proof: (1) It is easy to see that |= satisfies Autonomous Axioms 1-7, so |= is a rational consequence relation; so B |≈ A implies that B |= A.

(2) We establish "B |= A implies B |≈ A" by supposing B |≈ A *fails to hold*, and constructing, via a Henkin-proof-like method, that some truth-value assignment must make B true and A false. Thus, B |= A *fails to hold*. Here is how that works.

Suppose B |/≈ A. Then for some particular relation $|{\sim}_0$ that satisfies the Autonomous Axioms we have B $|/{\sim}_0$ A.

Define truth-set$_1$ to be {¬A}. Define $|{\sim}_1$ to be the relation such that for all sentences X, Y in the language of $|{\sim}_0$, X $|{\sim}_1$ Y just in case X·¬A $|{\sim}_0$ Y. By (RM), whenever B $|{\sim}_0$ C we have B·¬A $|{\sim}_0$ C, so B $|{\sim}_1$ C; also B·¬A $|/{\sim}_0$ A (if it did, then since B·A $|{\sim}_0$ A, we'd have B $|{\sim}_0$ A (WOR), contradiction!), so B $|/{\sim}_1$ A; and it's also easy to verify the $|{\sim}_1$ satisfies the Autonomous Axioms (using the additional rules derived above for the Autonomous Axioms).

Let **S** be some recursive enumeration of all the sentence in the language of $|{\sim}_0$, where this enumeration begins with sentence A. We inductively define a sequence of truth-sets$_k$ and *rational consequence relations* $|{\sim}_k$ based on considering each sentence k in the enumeration. We take $|{\sim}_1$ and truth-set$_1$ defined above as the basis of the induction.

Induction hypothesis: Suppose: (i) B $|/{\sim}_k$ ¬B; (ii) $|{\sim}_k$ satisfies the Autonomous Axioms 1-7; (iii) for all Z in truth-set$_k$, X $|{\sim}_k$ Z for every sentence X in the language.

Induction Step: Let C be sentence k+1.

If B $|{\sim}_k$ C, define X $|{\sim}_{k+1}$ Y to hold just when X·C $|{\sim}_k$ Y, and let truth-set$_{k+1}$ be the union of truth-set$_k$ with {C}.

If B $|/{\sim}_k$ C, then B $|/{\sim}_k$ ¬¬C: define X $|{\sim}_{k+1}$ Y to hold just when X·¬C $|{\sim}_k$ Y, and let truth-set$_{k+1}$ be the union of truth-set$_k$ with {¬C}.

(i) Notice that B $|/{\sim}_{k+1}$ ¬B.

(For, suppose B $|{\sim}_{k+1}$ ¬B.

Suppose B $|{\sim}_k$ C (for reductio): then B·C $|{\sim}_k$ ¬B, so B·C $|{\sim}_k$ ¬(C·B) (ADD, RC); and B·¬C $|{\sim}_k$ ¬C (R, SMP), so B·¬C $|{\sim}_k$ ¬(C·B) (ADD); so B $|{\sim}_k$ ¬(C·B) (WOR), then B $|{\sim}_k$ ¬(C·B)·C (AND), so B $|{\sim}_k$ ¬B (SYL) contra the induction hypothesis.

Thus B $|/{\sim}_k$ C: so B $|/{\sim}_k$ ¬¬C (DN), so from B $|{\sim}_{k+1}$ ¬B we have B·¬C $|{\sim}_k$ ¬B; but B·¬C $|{\sim}_k$ B (R, SMP), so B·¬C $|{\sim}_k$ C (CNTRA), B·C $|{\sim}_k$ C (R,SMP), so B $|{\sim}_k$ C (WOR), contradiction.)

(ii) Notice that $|{\sim}_{k+1}$ must satisfy the Autonomous Axioms 1-7 (using the additional rules derived above for the Autonomous Axioms), since $|{\sim}_k$ does.

(iii) Furthermore, notice that for all Z in truth-set$_{k+1}$, X $|{\sim}_{k+1}$ Z for every sentence X in the language, because:

(I) if C was added to truth-set$_k$ to get truth-set$_{k+1}$, then X·C $|{\sim}_k$ C and X·C $|{\sim}_k$ Z for all Z in truth-set$_k$, for all X in the language;

(II) if ¬C was added to truth-set$_k$ to get truth-set$_{k+1}$, then X·¬C $|{\sim}_k$ ¬C and X·¬C $|{\sim}_k$ Z for all Z in truth-set$_k$, for all X in the language.

Now, define *truth-set* to be the union of all the truth-set$_k$ for all k. *truth-set* is effectively a truth-value assignment. For, (i) *truth-set* must contain each sentence or its negation (since we went through an enumeration of all sentence and added a sentence or its negation at each step); (ii) *truth-set* cannot contain both a sentence and its negation (for, if it did, that would have first occurred at some step k+1, so truth-set$_k$ would already have contained either sentence C or its negation, before step k+1; suppose it already contained C; but by the construction we would already have had X $|{\sim}_k$ C and X $|/{\sim}_k$ ¬C for all X; so C would have been added to truth-set$_{k+1}$ rather than ¬C); and, (iii) *truth-set* contains (C·D) just in case it contains C and contains D: for,

(I) If truth-set$_k$ already contained C and D when (C·D) comes up in the enumeration at step k+1, then because AND holds we would already have B $|{\sim}_k$ (C·D), so (C·D) would be added to truth-set$_{k+1}$.

(II) If truth-set$_k$ already contained (C·D) when C (or D) comes up in the enumeration at step k+1, then because (SMP) holds, we would already have B $|{\sim}_k$ C (or B $|{\sim}_k$ D), so C (or D) would be added to truth-set$_{k+1}$.

(III) If truth-set$_k$ already contained ¬C (or ¬D) when (C·D) comes up in the enumeration at step k+1, then because (ADD) (and (RNC)) hold, we would already have B $|{\sim}_k$ ¬(C·D), so ¬(C·D) would be added to truth-set$_{k+1}$.

(IV) If truth-set$_k$ already contained C and ¬(C·D) when D comes up in the enumeration at step k+1, then because AND holds, we would already have B |~$_k$ ¬(C·D)·C, so (via SYL) B |~$_k$ ¬D, so ¬D would be added to truth-set$_{k+1}$. (Similarly when D and ¬(C·D) is already in the truth-set$_k$.)

Thus, *truth-set* supplies a truth-value assignment (i.e., assign *true* to all and only its members). This true-value assignment makes B true (it has to contain B, since when we come to sentence B we must already have B |~$_k$ B) and it makes ¬A true and A false (because at each step truth-set$_k$ contains ¬A).

Theorem 6: Weak Axioms 2 and 3 follow from the Autonomous Axioms.

proof: Suppose the Autonomous Axioms hold.

(1) If B |= A and C |~ B then C |~ A.

Suppose B |= A and C |~ B. Then |= ¬(B·¬A), so C |= ¬(B·¬A), so C |≈ ¬(B·¬A) (Tm 5), so C |~ ¬(B·¬A). Then, since we've already derived AND from the Autonomous Axioms, C |~ ¬(B·¬A)·B, so C |~ ¬¬A (SYL), so C |~ A (DN).

(2) If C |= B, B |= C, B |~ A, then C |~ A.

Suppose C |= B, B |= C, B |~ A. Then C |≈ B and B |≈ C (Tm5), so C |~ B and B |~ C, so C |~ A (LRE, Tm 1).

Theorem 7: The Autonomous axioms are inter-derivable with the Weak Axioms and with the Usual Axioms. The *Autonomous rational consequence relations* are just the *Weak rational consequence relations*, which are just the *Usual rational consequence relations*.

proof: The axioms are the same except for 2 and 3. It's easy to derive Autonomous Axioms 2 (LC), 3.1 (SMP-Left) and 3.2 (SMP-Right), 3.3 (DN), and 4.4 (SYL) from the Weak Axioms. Theorem 6 shows that Weak Axioms 2 (LCE) and 3(RW) are derivable from the Autonomous Axioms. Thus, the Autonomous Axioms are inter-derivable with the Weak Axioms, and Theorem 4 shows that the Weak Axioms are inter-derivable with the Usual Axioms.

Theorem 8: Every *truth-table consequence relation* is a *rational consequence relation*.

proof: Let |~$_T$ be the *truth-table consequence relation* for ranked truth-table *T*. We only need show that |~$_T$ has to satisfy the Weak Axioms for *rational consequence relation*. We go through the axioms one at a time.

(0) For atomic sentence A, a rank-1 line makes A∨¬A true and fails to make A·¬A true, so A∨¬A |/~$_T$ A·¬A.

(1) If B |= A and C |~$_T$ B, then every rank C line that makes C true must make B true, and so must make A true as well.

(3) If B |= C and C |= B, then B and C are made true by precisely the same truth-table lines. So all and only the rank C lines that make C true are rank B lines that make B true. Then, when B |~$_T$ A, every ranks B line that makes B true makes A true, so every rank C line that makes C true makes A true.

(4) Suppose (B·C) |~$_T$ A and (B·¬C) |~$_T$ A. Then either (i) the rank of B·C is higher than the rank of B·¬C, or (ii) the rank of B·¬C is higher than the rank of B·C, or (iii) B·C has the same rank as B·¬C. In case (i) all the rank B·C lines that make B·C true must also make B true, and all rank B lines that make B true must make B·C true (each rank B line is either a rank B·C line or a rank B·¬C line, or both, but in this case all B·¬C lines are at a lower rank than B·C lines; so all the rank B lines that make B true also make B·C true); thus, the rank B·C lines that make B·C true are precisely the rank B lines that make B true; so, form (B·C) |~$_T$ A it follows that all rank B lines that make B true must make A true as well. Case (ii) is similar, but with '¬C' interchanged with C. In case (iii), the rank of B can be no higher than that of either B·C or B·¬C, and must be at least as high as one of them, so it must be the same rank as both of them; and the set of rank B lines that make B true is just the union of the rank B lines that make B·C true with the set of rank B lines that make B·¬C true; so from (B·C) |~$_T$ A and (B·¬C) |~$_T$ A we have that all rank B lines that make B true must make A true, B |~$_T$ A.

(5) Suppose B |~$_T$ (C·A). So all rank B lines that make B true make C true (so B·C must be at the same rank as B), and those same lines make A true; so all rank B·C lines that make B·C true must make A true.

(6.1) Suppose (C·B) |~$_T$ A. Then all rank C·B lines that make C·B true make C true and B true and A true; so all rank C·B lines that make C·B true make B·A true as well.

(6.2) Suppose B $|\!\sim_T$ A and B $|\!\sim_T$ ¬A. Then each rank B line that makes B true must make both A true and make ¬A true. No finite (non-ω) ranked line can do that. So the rank of B is ω. And when B has rank ω, B $|\!\sim_T$ C for all C.

(7) Suppose B $|\!\sim_T$ A and B $|\!/\!\!\sim_T$ ¬C. Then all rank B lines that make B true make A true, and not all of these lines make ¬C true. Thus, some rank B lines that make B true make B·C true; and all of the rank B lines that make B·C true must be rank B·C lines (since B cannot be true at any higher rank than rank B, so B·C cannot be true at any higher rank than rank B). Thus, since all rank B lines that make B true must also make A true, all rank B·C lines that make B·C true must make A true.

3. Section 5

Definition 10: *Usual Axioms* for the *Popper functions*: Let *L* be a language having the syntax of sentential logic. A *Popper function* is any function P from pairs of sentences of *L* to the real numbers such that:

0. For some E, F, G, H, P[E | F] ≠ P[G | H] (non-triviality)
1. P[A | B] ≥ 0 (Non-negativity)
2. If C |= B, B |= C, then P[A | B] = P[A | C] (right logical equivalence)
3. If B |= A, then P[A | B] = 1 (logical entailment)
4. If C |= ¬(B·A), then P[(A∨B) | C] = P[A | C] + P[B | C] or P[D | C] = 1 (additivity)
5. P[(A·B) | C] = P[A | (B·C)] × P[B | C] (conditionalization)

Definition 11: *Autonomous Axioms* for the *Popper functions*: Let *L* be a language having the syntax of sentential logic. An *autonomous Popper function* is any function P from pairs of sentences of *L* to the real numbers (not necessarily restricted between 0 and 1) such that:

0. for some E, F, G, H, P[F | E] ≠ P[G | H] (non-triviality)

and for all sentences A, B, C,

1. P[A | A] ≥ P[B | B] (self-support)
2. P[A | (B·C)] ≥ P[A | (C·B)] (right commutivity)
3. P[A | C] ≥ P[(A·B) | C] (left simplification)
4. P[A | B] + P[¬A | B] = P[B | B] or P[D | B] = P[B | B] for all D (weak additivity)
5. P[(A·B) | C] = P[A | (B·C)] × P[B | C] (conditionalization)

Theorem 9: The usual axioms for Popper functions imply that:
(i) 1 ≥ P[A | B] ≥ 0
(ii) If B |= A, then P[A | C] ≥ P[B | C]

proof: (i) Axiom 1 already provides P[A | B] ≥ 0.
Suppose P[A | B] > 1. Then from axioms 2, 4, and 1 we derive a contradiction:
1 = P[A∨¬A | B] = P[A | B] + P[¬A | B], so 0 > 1 − P[A | B] = P[¬A | B] ≥ 0, i.e. 0 > 0.
(ii) Suppose B |= A. Suppose P[D | C] < 1 for at least some D (otherwise P[A | C] = 1 ≥ P[B | C] and we are done). From B |= A we get |= ¬B∨(B·A), so C |= ¬B∨(B·A); also |= ¬(¬B·(B·A)), so C |= ¬(¬B·(B·A)); then (by axioms 3 and 4) 1 = P[¬B∨(B·A) | C] = P[¬B | C] + P[(B·A) | C]; from C |= B∨¬B and C |= ¬(B·¬B) (by axioms 3 and 4) we have P[B | C] = 1 − P[¬B | C]. Thus, P[B | C] = 1 − P[¬B | C] = P[(B·A) | C] = P[B | A·C] × P[A | C] ≤ P[A | C].

Theorem 10: The Autonomous Axioms for *Popper functions* imply 1 = P[B | B] ≥ P[A | B] ≥ 0.

proof: For all A, B, P[A | A] = P[B | B] = k for some real k (since by axiom 1, for some k, k = P[A | A] ≥ P[B | B] ≥ P[A | A] = k). Notice that (from axioms 3 and 5) k = P[A | A] ≥ P[(A·A) | A] ≥ P[(A·A)·A | A] = P[(A·A) | (A·A)] × P[A | A] = k^2. But for any real k, $k^2 \geq 0$; so $k \geq k^2 \geq 0$, so 1 ≥ k ≥ 0. Then axiom 4 yields: either k = P[B | B] + P[¬B | B] = k + P[¬B | B] or P[D | B] = k for all D; so either P[¬B | B] = 0 or P[A | B] = k; then (by axioms 3, 4) P[A | B] ≥ P[A·¬B | B] = P[A | ¬B·B] × P[¬B | B] = 0 or P[A | B] = k ≥ 0; thus P[A | B] ≥ 0 (for all A, B).

Then (for all A, B) P[¬A | B] ≥ 0 and P[A | B] ≥ 0, so P[A | B] = k or k = P[A | B] + P[¬A | B] ≥ P[A | B]; thus k ≥ P[A | B] ≥ 0 (for all A, B). Then (from axiom 3) P[A | A·A] = k (since k ≥ P[A | A·A] = P[A·A | A·A] = k) and P[A | A·(A·A)] = k (since k ≥ P[A | A·(A·A)] ≥ P[A·(A·A) | A·(A·A)] = k), so k = P[A·A | A·A] = P[A |

A·(A·A)] × P[A | (A·A)] = k^2; then k = k^2 ≥ 0, so either k = 0 or 1 = k. But if k = 0 we have (for all A, B) 0 = k ≥ P[A | B] ≥ 0, so P[A | B] = 0 (for all A, B); this contradicts axiom 0. Thus, 1 = k = P[B | B] ≥ P[A | B] ≥ 0.

Theorem 11: The Autonomous Axioms imply that P[A·B | C] = P[B·A | C] (for all A, B, C).

proof: 1 = P[(A·(B·C)) | (A·(B·C))] = P[A | (B·C)·(A·(B·C))] × P[(B·C) | (A·(B·C))], so 1 = P[(B·C) | (A·(B·C))], so 1 ≥ P[B | (A·(B·C))] ≥ P[(B·C) | (A·(B·C))] = 1. Thus, P[B | (A·(B·C))] = 1 = P[(B·C) | (A·(B·C))]. Then P[B·A | (B·C)] = P[B | A·(B·C)] × P[A | (B·C)] = P[A | (B·C)]; then P[(B·A) | C] ≥ P[(B·A)·B | C] = P[B·A | (B·C)] × P[B | C] = P[A | (B·C)] × P[B | C] = P[A·B | C]; thus P[(B·A) | C] ≥ P[(A·B) | C]. Similarly, P[(A·B) | C] ≥ P[(B·A) | C]. So P[(A·B) | C] = P[(B·A) | C].

Theorem 12: For each P that satisfies the Autonomous Axioms for the *autonomous Popper functions*, consider the associated relation |~ defined as B |~ A *just when* P[A | B] = 1. Then |~ satisfies the Autonomous Axioms for the *rational consequence relations*; thus $|\sim_P$ is a *rational consequence relation*.

proof: Let P be any function that satisfies the Autonomous Axioms for *Popper functions* and define the relation |~ as follows: B |~ A *just when* P[A | B] = 1. We show that the Autonomous Axioms for the *rational consequence relations* must hold for |~.

The derivations of Autonomous Axioms 0-3.2 for the *rational consequence relations* from the Autonomous *Popper function* axioms are straightforward (given the results already proved for the Autonomous Popper Functions). Here are derivations of the rest.

3.3 Suppose B |~ ¬¬A. If P[D | B] = 1 for all D, then P[A | B] = 1, so B |~ A and we are done. So suppose, for some D, P[D | B] < 1. Then 1 = P[¬¬A | B] = 1 − P[¬A | B] = 1 − (1 − P[A | B]) = P[A | B], so B |~ A.

3.4 Suppose C |~ (¬(B·A)·B). If P[D | C] = 1 for all D, then P[¬A | C] = 1, so C |~ ¬A and we are done. So suppose, for some D, P[D | C] < 1. Then 1 = P[(¬(B·A)·B) | C] = P[B·¬(B·A) | C] ≤ P[B | C] ≤ 1, so P[B | C] = 1; and 1 ≥ P[¬(B·A) | C] ≥ P[¬(B·A)·B | C] = 1, so 1 = P[¬(B·A) | C] = 1 − P[B·A | C]. Then 0 = P[B·A | C] = P[A·B | C] = P[A | B·C] × P[B | C] = P[A | B·C] ≥ 0, so 0 = P[A | B·C] = 1 − P[¬A | B·C], so P[¬A | B·C] = 1. Then 1 = P[¬A | B · C] × P[B | C] = P[¬A·B | C] = P[B·¬A | C] = P[B | ¬A·C] × P[¬A | C], so P[¬A | C] = 1, thus C |~ ¬B.

4. Suppose B·C |~ A, B·¬C |~ A. Then P[A | B·C] = 1 and P[A | B·¬C] = 1. If P[D | B] = 1 for all D, then P[A | B] = 1, so B |~ A and we are done. So suppose that for some D, P[D | B] < 1. Then P[A | C·B] = P[A | B·C] = 1 and P[A | ¬C·B] = P[A | B·¬C] = 1. So P[C | A·B] × P[A | B] = P[C·A | B] = P[A·C | B] = P[A | C·B] × P[C | B] = P[C | B] and P[¬C | A·B] × P[A | B] = P[¬C·A | B] = P[A·¬C | B] = P[A | ¬C·B] × P[¬C | B] = P[¬C | B] = 1 − P[C | B]. Then, 1 = P[C | B] + P[¬C | B] = (P[C | A·B] × P[A | B]) + (P[¬C | A·B] × P[A | B]) = P[A | B] × (P[C | A·B] + P[¬C | A·B]); thus, 1 = P[A | B] × (P[C | A·B] + P[¬C | A·B]). Now either P[D | A·B] = 1 for all D or else P[C | A·B] + P[¬C | A·B] = 1; so either P[C | A·B] + P[¬C | A·B] = 2 or else P[C | A·B] + P[¬C | A·B] = 1; so P[C | A·B] + P[¬C | A·B] ≥ 1; so 1 = P[A | B] × (P[C | A·B] + P[¬C | A·B]) ≤ P[A | B] ≤ 1; so P[A | B] = 1. Then, B |~ A.

5. Suppose B |~ (C·A), then 1 = P[C·A | B] = P[A·C | B] = P[A | C·B] × P[C | B], so 1 = P[A | C·B] = P[A | B·C], so B·C |~ A.

6.1 Suppose (C·B) |~ A. Then P[A | C·B] = 1, so P[B·A | (C·B)] = P[B | A·(C·B)] × P[A | C·B] = P[B | A·(C·B)], and 1 = P[A·(C·B) | A·(C·B)] = P[(C·B)·A | A·(C·B)] ≤ P[(C·B) | A·(C·B)] = P[(B·C) | A·(C·B)] ≤ P[B | A·(C·B)] ≤ 1, so 1 = P[B | A·(C·B)] = P[B | A·(C·B)] × P[A | C·B] = P[B·A | (C·B)]; then C·B |~ B·A.

6.2 Suppose B |~ A, B |~ ¬A. Then P[A | B] = 1 = P[¬A | B], so 2 = P[A | B] + P[¬A | B] > 1, so P[C | B] = 1; thus B |~ C.

7. Suppose B |~ A, B |/~ ¬C. Then P[¬C | B] < 1, so P[C | B] = 1 − P[¬C | B] > 0 , and 1 = P[A | B] = 1 − P[¬A | B]; so 0 = P[¬A | B] = P[C | ¬A·B] × P[¬A | B] = P[C·¬A | B] = P[¬A·C | B] = P[¬A | C·B] × P[C | B], so 0 = P[¬A | C·B] = 1 − P[A | C·B], so 1 = P[A | C·B] = P[A | B·C]; thus B·C |~ A.

Definition 12: pf-logical entailment, B |-- A:

'B |-- A' abbreviates "for every function P that satisfies the Autonomous Axiom, P[A | B] = 1".
Read 'B |-- A' as "B pf-entails A" – i.e. "B Popper-function-entails A".

Theorem 13: B |-- A if and only if B |= A.

proof: (i) Suppose B |= A. Then B |≈ A – i.e., for each rational consequence relation, B |~ A holds. Let P be any *autonomous Popper function* (i.e. any function that satisfies the Autonomous Axioms). Then the relation $|\sim_P$, defined as X $|\sim_P$ Y just in case P[Y | X] = 1, is a *rational consequence relation*; so B $|\sim_P$ A; so P[A | B] = 1. So, for each P that satisfies the Autonomous Axioms, P[A | B] = 1. Thus, B |--A.

(ii) Suppose *not* B |= A. Then there is a truth-value assignment that makes B true and A false – i.e. a truth-value assignment T that makes (B ⊃ A) false. Define a function T[Y | X] such that: (1) T[Y | X] = 1 whenever truth-value assignment T makes (X ⊃ Y) true; (2) T[Y | X] = 0 whenever T makes (X ⊃ Y) false. It's easy to check that the function T[Y | X] satisfies the Autonomous Axioms for *Popper functions*. But T[A | B] = 0 ≠ 1. So there is a Popper function P such that P[A | B] ≠ 1. Then *not* B |-- A.

Theorem 14: The functions that satisfy the Autonomous Axioms for Popper functions are identical to the functions that satisfy the Usual Axioms for Popper functions.

proof: (i) If P satisfies the Usual Axioms for *Popper functions*, then P satisfies the Autonomous Axioms. For ... All of the Autonomous Axioms are obviously derivable from the Usual Axioms, except perhaps axiom 3. But the Usual Axioms imply that whenever B |= A, P[A | C] ≥ P[B | C]; Autonomous Axiom 3 follows immediately from this.

(ii) We show that the Usual Axioms are derivable from the Autonomous Axioms (so every function that satisfies the Autonomous Axioms must also satisfy the Usual Axioms). This is obvious for Usual Axioms 0, 1 (via Tm 10), and 5. We derive the others here.

2. Suppose that C |= B and B |= C. Then A·C |= B and A·B |= C, so then C |-- B, B |-- C, A·C |-- B, A·B |-- C. Let P be an Autonomous *Popper function* (i.e. a function that satisfies the Autonomous Axioms). Then P[B | C] = 1 = P[C | B] and P[B | A·C] = 1 = P[C | A·B] (Tm 13). Then (using only Autonomous Axioms and things we've already derived from them) P[A | B] = P[C | A·B] × P[A | B] = P[C·A | B] = P[A·C | B] = P[A | C·B] × P[C | B] = P[A | C·B] = P[A | B·C] = P[A | B·C] × P[B | C] = P[A·B | C] = P[B·A | C] = P[B | A·C] × P[A | C] = P[A | C].

3. Suppose that B |= A. Then B |-- A, so P[A | B] = 1.

4. Suppose that C |= ¬(B·A). We can suppose that for some D, P[D | C] ≠ 1 (otherwise, for all D, P[D | C] = 1, which is one disjunct of the conclusion to be derived, which completes the derivation). From C |= ¬(B·A) we have C |= ¬(A·B), so C |-- ¬(A·B), so P[¬(A·B) | C] = 1, so P[A·B | C] = 0.

We must have *either*, for some D, P[D | ¬A·C] ≠ 1, *or* for some D, P[D | ¬B·C] ≠ 1 – otherwise we get a contradiction, as follows:

Suppose for all D P[D | ¬A·C] = 1 and for all D P[D | ¬B·C] = 1. Then P[¬C | ¬A·C] = 1 and P[¬C | ¬B·C] = 1, so 0 = 1 – P[C | C] = P[¬C | C] ≥ P[¬C·¬A | C] = P[¬C | ¬A·C] × P[¬A | C] = P[¬A | C] and 0 = 1 – P[C | C] = P[¬C | C] ≥ P[¬C·¬B | C] = P[¬C | ¬B·C] × P[¬B | C] = P[¬B | C]; so P[¬A | C] = 0 and P[¬B | C] = 0, and P[B | C] = 1. So, 0 = P[A·B | C] = P[A | B·C] × P[B | C] = P[A | B·C] = 1 – P[¬A | B·C] = 1 – (P[¬A | B·C] × P[B | C]) = 1 – P[¬A·B | C] = 1 – P[B·¬A | C] = 1 – P[B | ¬A·C] × P[¬A | C] = 1, so 0 = 1, contradiction.

With no loss of generality, suppose it's for some D, P[D | ¬B·C] ≠ 1.

By the definition of '∨' in terms of '¬' and '·'we have

P[A∨B | C] = P[¬(¬A·¬B) | C] = 1 – P[¬A·¬B | C] = 1 – P[¬A | ¬B·C] × P[¬B | C] = 1 – (1 – P[A | ¬B·C]) × P[¬B | C] = 1 – P[¬B | C] + P[A·¬B | C] = P[B | C] + P[¬B | A·C] × P[A | C]. Then *either* P[¬B | A·C] = 1 and so P[A∨B | C] = P[A | C] + P[B | C] and we are done, *or* P[¬B | A·C] ≠ 1 and so P[A∨B | C] = P[B | C] + (1 – P[B | A·C]) × P[A | C] = P[B | C] + P[A | C] – P[A·B | C] = P[A | C] + P[B | C].

Theorem 15: For each probabilistic ranked truth-table, the associated truth-table probability function is a *Popper function*.

proof: It's easy to check that each truth-table probability function must satisfy the Usual Axioms for *Popper functions*.

4. Section 6

Definition 13: The Rank-Orderings of Sentences Imposed by *Rational Consequence Relations*: For each *rational consequence relation* |~, define the relation $\geq_{|\sim}$ on sentences of its language as follows:

‘A $\geq_{|\sim}$ B’ abbreviates “A∨B |/~ ¬A or A∨B |~ ¬B”;
read ‘A $\geq_{|\sim}$ B’ as “the rank of A is at least as high as the rank of B for |~.”

1. ‘A $\approx_{|\sim}$ B’ abbreviates A $\geq_{|\sim}$ B and B $\geq_{|\sim}$ A;
   read ‘A $\approx_{|\sim}$ B’ as “A and B have the same rank for |~.”
2. ‘A $>_{|\sim}$ B’ abbreviates “A $\geq_{|\sim}$ B and *not* B $\geq_{|\sim}$ A”;
   read ‘A $>_{|\sim}$ B’ as “*the rank of* A *is higher than the rank of* B *for* |~.”
3. By definition, B has rank-ω for |~ just when B |~ ¬B.
4. By definition, B has rank-1 for |~ just when (C∨¬C) |/~ ¬B.
5. By definition, A is a rank B sentence for |~ just when A $\approx_{|\sim}$ B.

Theorem 16: From these definitions it follows immediately that:

1. A $\approx_{|\sim}$ A.
2. if A $\approx_{|\sim}$ B, then B $\approx_{|\sim}$ A.
3. A $>_{|\sim}$ B if and only if *not* B $\geq_{|\sim}$ A.
4. If |= D, then D is rank-1 – rank-1 is the rank of tautologies.
5. If |= D, then D $\geq_{|\sim}$ E for all E.
6. B is rank-1 if and only if, for some D such that |= D, B $\approx_{|\sim}$ D.
7. If A and B are both rank-1, then A $\approx_{|\sim}$ B.
8. If |= ¬D, then D is rank-ω – rank-ω is the rank of contradictions.
9. If |= ¬D, then E $\geq_{|\sim}$ D for all E.
10. B is rank-ω if and only if, for some D such that |= ¬D, B $\approx_{|\sim}$ D.
11. If A and B are both rank-ω, then A $\approx_{|\sim}$ B.
12. (A∨B |~ ¬B and A∨B |~ ¬A) if and only if A and B are both rank-ω.
13. If A and B are not both rank-ω, then
    13.1. A $\geq_{|\sim}$ B if and only if A∨B |/~ ¬A;
    13.2. A $\approx_{|\sim}$ B if and only if A∨B |/~ ¬A and A∨B |/~ ¬B;
    13.3. A $>_{|\sim}$ B if and only if A∨B |/~ ¬A and A∨B |~ ¬B.
14. If B |= A, then A $\geq_{|\sim}$ B.
15. If A $\geq_{|\sim}$ B, then A∨B $\approx_{|\sim}$ A.

proof: All of these follow quickly from Definition 3 together with easily derived properties of *rational consequence relations*. 14 and 15 are a bit tricky, so I provide their derivations.

14. Suppose B |= A. Then B |~ B (R), so B |~ A (RW) and A |~ A (R), so A∨B |~ A (OR). Then either A∨B |/~ ¬A (thus A $\geq_{|\sim}$ B), or else: A∨B |~ ¬A, so A∨B |~ A·¬A (AND), so A∨B |~ ¬B (RW), then A $\geq_{|\sim}$ B.

15. Suppose A $\geq_{|\sim}$ B. (i) Then A∨B |/~ ¬A or A∨B |~ ¬B, so A∨(A∨B) |/~ ¬A or A∨(A∨B) |~ ¬B (LCE), so A∨(A∨B) |/~ ¬A or [A∨(A∨B) |~ ¬A and A∨(A∨B) |~ ¬B], so A∨(A∨B) |/~ ¬A or A∨(A∨B) |~ ¬A·¬B (AND), so A∨(A∨B) |/~ ¬A or A∨(A∨B) |~ ¬(A∨B), then A $\geq_{|\sim}$ A∨B.
(ii) A |= A∨B, so A∨B $\geq_{|\sim}$ A (14 above). Thus, A∨B $\approx_{|\sim}$ A.

Theorem 17: Completeness: A $\geq_{|\sim}$ B or B $\geq_{|\sim}$ A.

proof: Suppose *not* A $\geq_{|\sim}$ B. Then *not* [A∨B |/~ ¬A or A∨B |~ ¬B], so A∨B |~ ¬A and A∨B |/~ ¬B. Then, clearly, either A∨B |~ ¬A or A∨B |/~ ¬B. Thus, by definition, B $\geq_{|\sim}$ A.

Theorem 18: Transitivity: If A $\geq_{|\sim}$ B and B $\geq_{|\sim}$ C, then A $\geq_{|\sim}$ C.

proof: Suppose A $\geq_{|\sim}$ B and B $\geq_{|\sim}$ C – i.e. suppose [A∨B |/~ ¬A *or* {A∨B |~ ¬A and A∨B |~ ¬B}] and [B∨C |/~ ¬B *or* { B∨C |~ ¬B and B∨C |~ ¬C}]. Then [A∨B |/~ ¬A *or* {A∨B |~ ¬A·¬B, so A∨B |~ ¬(A∨B)·(A∨B) }] and [B∨C |/~ ¬B *or* {B∨C |~ ¬B·¬C, so B∨C |~ ¬(B∨C)·(B∨C)}]. Then, [A∨B |/~ ¬A *or* {A∨B |~ A·(A·¬A) and A∨B |~ B·(A·¬A)}] and [B∨C |/~ ¬B *or* {B∨C |~ B·(A·¬A) and B∨C |~ C·(A·¬A)}]. Then, [A∨B |/~ ¬A *or* {(A∨B)·A |~ (A·¬A) and (A∨B)·B |~ (A·¬A)}] and [B∨C |/~ ¬B *or* {(B∨C)·B |~ (A·¬A) and (B∨C)·C |~ (A·¬A)}]. Thus, [A∨B |/~ ¬A *or* {A |~ (A·¬A) and B |~ (A·¬A)}] and [B∨C |/~ ¬B *or* {B |~ (A·¬A) and C |~ (A·¬A)}].

So: either (i) A∨B |/~ ¬A and B∨C |/~ ¬B;
or (ii) A∨B |/~ ¬A and B |~ (A·¬A) and C |~ (A·¬A);
or (iii) B∨C |/~ ¬B and A |~ (A·¬A) and B |~ (A·¬A);
or (iv) A |~ (A·¬A) and B |~ (A·¬A) and C |~ (A·¬A).

Suppose (for reductio) *not* A $\geq_{|\sim}$ C. Then *not* [A∨C |/~ ¬A or A∨C |~ ¬C].

So A∨C |~ ¬A and A∨C |/~ ¬C. We show that each of (i) through (iv) leads to a contradiction; and that will complete the reductio proof.

Suppose (i). Then, from A∨C |~ ¬A, ((A∨B)∨C)·(A∨C) |~ ¬A (LCE), and ((A∨B)∨C)·¬(A∨C) |= ¬A, so ((A∨B)∨C)·¬(A∨C) |~ ¬A (R,RW), so ((A∨B)∨C) |~ ¬A (by WOR). If ((A∨B)∨C) |/~ ¬(A∨B), then (RM) ((A∨B)∨C)·(A∨B) |~ ¬A, so (A∨B) |~ ¬A (LCE), contradiction with (i). Thus, ((A∨B)∨C) |~ ¬(A∨B), so ((A∨B)∨C) |~ ¬B (RW). If ((A∨B)∨C) |/~ ¬(B∨C), then by RM ((A∨B)∨C)·(B∨C) |~ ¬B, so (B∨C) |~ ¬B (LCE), contradiction with (i). Thus, ((A∨B)∨C) |~ ¬(B∨C), so ((A∨B)∨C) |~ ¬C (RW). Thus, ((A∨B)∨C) |~ ((¬A·¬B)·¬C) (two applications of AND). Then ((A∨B)∨C) |~ ¬((A∨B)∨C) (RW) and ((A∨B)∨C) |~ ((A∨B)∨C) (R), so ((A∨B)∨C) |~ ((A∨B)∨C)·¬((A∨B)∨C) (AND), so ((A∨B)∨C) |~ (A∨B)·¬A (RW), so ((A∨B)∨C)·(A∨B) |~ ¬A (VCM), so (A∨B) |~ ¬A, contradiction with (i).

Suppose (ii). From C |~ (A·¬A) we get C |~ (A∨C)·¬C (RW), so C·(A∨C) |~ ¬C (VCM), so (A∨C)·C |~ ¬C (LCE). Also, (A∨C)·¬C |~ (A∨C)·¬C (R), so (A∨C)·¬C |~ ¬C (RW). Then (A∨C) |~ ¬C (WOR), but we also have (A∨C) |/~ ¬C, (directly above "Suppose (i)") contradiction.

Suppose (iii). From B |~ (A·¬A) we get B |~ (B∨C)·¬B (RW), so B·(B∨C) |~ ¬B (VCM), so (B∨C)·B |~ ¬B (LCE). Also, (B∨C)·¬B |~ (B∨C)·¬B (R), so (B∨C)·¬B |~ ¬B (RW). Then (B∨C) |~ ¬B (WOR), but we also have (B∨C) |/~ ¬B (iii), contradiction.

Suppose (iv). Exactly the same argument as for (ii), leading to a contradiction.

Theorem 19: For each *rational consequence relation* |~, its associated relation $\geq_{|\sim}$ is a *total preorder* on the sentences of its language (i.e. a complete, transitive relation); the associated relation $\approx_{|\sim}$ is an *equivalence relation* (i.e. a reflexive, symmetric, transitive relation). The relation $\geq_{|\sim}$ yields a partition of the sentences of the language into an ordered hierarchy (ordered according to $>_{|\sim}$) of mutually exclusive and exhaustive equivalence classes (equivalent under relation $\approx_{|\sim}$) of sentences.

proof: Follows directly from Theorems 16-18.

Theorem 20: For each *rational consequence relation* |~ (on a finite or countably infinite language), B |~ A holds *just in case* either (i) B is at rank-ω, or (ii) in every finite sub-language *L* of the language of |~ that contains all sentence letter in A and B, every rank B state-description in *L* that logically entails B also logically entails A. (A state description in *L* is a conjunction consisting of every sentence letter in *L* or its negation, but not both).

proof: (I -- *only if* part): Let |~ be a *rational consequence relation*. Let A and B be any two sentences in a finite sub-language *L* of the language of |~, and suppose that B |~ A. Suppose B is not at rank-ω. Let S be a state-description in *L* that is at the rank of B and logically entails B. (We show that S |= A).

Since S is at the same rank as B, either they are both at rank-ω, or neither is at rank-ω and B∨S |/~ ¬S and B∨S |/~ ¬B. But we are suppose B not at rank-ω, so neither is at rank-ω, Notice that S |/~ ¬S (else, if S |~ ¬S, then (B∨S)·S |~ ¬S (LCE), and (B∨S)·¬S |~ ¬S (R, RW), so (B∨S) |~ ¬S (WOR), contradiction). Notice that (B∨S) is logically equivalent to B (since B |= B∨S, and B |= B and S |= B, so B∨S |= B). Then, since B |~ A, B∨S |~ A (LCE), so (B∨S)·S |~ A (RM), so S |~ A (LCE), so S |~ (S·A) (R, AND), so S·A is logically consistent (otherwise S |~ ¬S (RW), contra.), So B does not logically entail ¬A. But each state-description in the language containing A either logically entails A or logically entails ¬A; so S must logically entail A.

(II -- *if* part): Let |~ be a *rational consequence relation*. Let A and B be any two sentences in a finite sub-language *L* of the language of |~, and suppose that (i) B is at rank-ω, or (ii) B is not at rank-ω, and in every finite sub-language *L* of the language of |~ that contains all sentence letter in A and B, every rank B state-description in *L* that logically entails B also logically entails A. (We show that B |~ A.)

(i) Suppose that B is at rank-ω. Then B |~ ¬B (by definition of rank-ω) and B |~ B (R), so B |~ B·¬B, so B |~ A.

(ii) Suppose B is not at rank-ω, and in every finite sub-language *L* of the language of |~ that contains all sentence letter in A and B, every rank B state-description in *L* that logically entails B also logically entails A. Suppose (for reductio) B |/~ A. Then B |/~ (B·A).

Now, let *L* be any language that contains all sentence letters in A and B. Then B is logically equivalent to a disjunction of state-descriptions of *L*, say $(B_1\vee...\vee B_n)$. Each B_i must be rank B or lower

[else, for B_i at a higher rank than B, $B_i\vee B$ |/~ $\neg B_i$ and $B_i\vee B$ |~ ¬B, so $(B_i\vee B)\cdot B_i$ |~ ¬B (RM), so B_i |~ ¬B (LCE), and B_i |~ $(B_1\vee...\vee B_i\vee...\vee B_n)$ (R, RW (n−1) times), so B_i |~ B (RW), then B_i |~ $\neg B_i$ (AND, RW) and $\neg B_i\cdot B$ |~ $\neg B_i$ (R, RW), so $B_i\vee(\neg B_i\cdot B)$ |~ $\neg B_i$ (OR), so $B_i\vee B$ |~ $\neg B_i$ (LCE) contradiction].

With no loose of generality we may let $\{B_1, ..., B_k\}$ be the rank B state-descriptions in $(B_1\vee...\vee B_n)$, and $\{B_{k+1}, ..., B_n\}$ be lower ranking state-descriptions. $\{B_1, ..., B_k\}$ must not be empty (else, for all B_i in $(B_1\vee...\vee B_n)$, B |~ $\neg B_i$, so B |~ $\neg B_1\cdot...\cdot\neg B_n$ (AND n times), so B |~ $\neg(B_1\vee...\vee B_n)$ (RW), so B |~ ¬B (RW), so B has rank-ω, contra.). Now, since each rank B state-description that logically entail B also logically entails A, we must have $(B_1\vee...\vee B_k)$ |= A; so $(B_1\vee...\vee B_k)$ |~ A (R,RW).

$(B_1\vee...\vee B_k)$ has rank B (because, whenever C and D have the same rank above ω, C∨D has that same rank as C, since: C∨D |/~ ¬C and C∨D |/~ ¬D, so (C∨D) |/~ ¬C·¬D, so (C∨D) |/~ ¬(C∨D), so (C∨D)∨C |/~ ¬(C∨D) and (C∨D)∨C |/~ ¬C).

$(B_{k+1}\vee...\vee B_n)$ has lower rank than B (because, whenever D and E both have lower rank than C, C∨D |~ ¬D and C∨E |~ ¬E, so (C∨D) |~ ¬(D∨E) and (C∨E) |~ ¬(D∨E) (RW), so (C∨D)∨(C∨E) |~ ¬(D∨E) (OR), so C∨(D∨E) |~ ¬(D∨E) (LCE)).

Then, $(B_1\vee...\vee B_k)\vee(B_{k+1}\vee...\vee B_n)$ |~ $\neg(B_{k+1}\vee...\vee B_n)$, so $(B_1\vee...\vee B_k)\vee(B_{k+1}\vee...\vee B_n)$ |~ $(B_1\vee...\vee B_k)$ (R, AND, RW), and $(B_1\vee...\vee B_k)$ |~ $(B_1\vee...\vee B_k)\vee(B_{k+1}\vee...\vee B_n)$ (R, RW), and we already have $(B_1\vee...\vee B_k)$ |~ A, thus $(B_1\vee...\vee B_k)\vee(B_{k+1}\vee...\vee B_n)$ |~ A (LRE, Tm 16). Thus, B |~ A.

Theorem 21: Each *truth-table consequence relation* is a *rational consequence relation* (on a finite language). Each finite part of a *rational consequence relation* |~ (each part of |~ defined on a finite sub-language of the language of |~) is a *truth-table consequence relation*.

proof: We've already proved the first claim: Theorem 8.

To see that each finite part of a *rational consequence relation* |~ (the part of |~ defined on a finite sub-language) is a *truth-table consequence relation*:

Let *L* be a finite sublanguage of the language of |~. Order the state-descriptions into a ranked hierarchy of equivalence classes (equivalent according to $\approx_{|\sim}$), ranked in the order provided by $>_{|\sim}$. Construct ranked truth-table *T* by constructing a truth-table for *L*; assign rank-1 to just those the truth-table lines that make the highest-ranked (rank-1) state-descriptions true according to ordering $\geq_{|\sim}$; assign rank-2 to the truth-table lines that make the second highest ranked state-descriptions true according to $\geq_{|\sim}$; ...; assign rank-ω to those truth-table lines that make rank-ω state-descriptions true according to $\geq_{|\sim}$. The resulting ranked truth-table will provide a truth-table consequence relation ||~ that agrees with |~, due to the 1-1 relationship between truth-table lines and the state-descriptions that make them true; and the fact that a truth-table line makes a sentence B true *just when* the state-description that line makes true logically entails B; this also aligns the ranks of sentences, since the rank of B for |~ on *L* is the highest rank of the state-descriptions that entail B, and the rank of B for ||~ on *L* is the highest rank of the truth-table line that makes it true. Thus, Theorem 20 and the construction:

B |~ A *if and only if* "B has rank-ω *or* all rank B state-descriptions that logically entail B also logically entail A" *if and only if*

"B has rank-ω *or* all rank B truth-table lines that make B true also make A true".

Theorem 22: For each *rational consequence relation* |~ there is a *Popper function* P such that for all sentences A, B in the language of |~, P[A | B] = 1 just when B |~ A.

proof: Given a rational consequence relation |~, we construct a *Popper function* $P_{|\sim}$ for |~ such that P[A | B] = 1 just when B |~ A. To do this, first specify the sentence letters of the language in some "alphabetical order". Then, for each n, consider the finite language L_n for sentential logic that is based on only the first n sentence letters. Construct the set of state-descriptions, S_n, for L_n. where each state-description in S_n has its sentences letters (or their negations) in the alphabetical order for sentence letters. For set S_n of state-descriptions, find the rank of each of its members according to |~, and assign it a weight $w_{q,n}$ within its rank, *q*, in the following way:

A member of set S_n has form $A_1\cdot\ldots\cdot A_{n-1}\cdot A_n$, where each A_k is a sentence letter or its negation, and where A_n is either the "new" sentence letter (i.e. the sentence letter not in L_{n-1}) or the negation of the "new" sentence letter. Call that "new" sentence letter 'C'. Thus, $A_1\cdot\ldots\cdot A_{n-1}\cdot A_n$ is of form $A_1\cdot\ldots\cdot A_{n-1}\cdot C$ or $A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C$, where $A_1\cdot\ldots\cdot A_{n-1}$ is a state description from set S_{n-1} for language L_{n-1}. Notice that $A_1\cdot\ldots\cdot A_{n-1}$ is logically equivalent to $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$. So the rank of $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$ is the rank of $A_1\cdot\ldots\cdot A_{n-1}$.

If $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \mid\!\sim \neg(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$ and $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \not\mid\!\sim \neg(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$, then $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$ will be at the same rank, q, as $(A_1\cdot\ldots\cdot A_{n-1})$, and $(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$ will be at some lower rank u. Assign $w_{q,n}[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = w_{q,n-1}[A_1\cdot\ldots\cdot A_{n-1}]$, $w_{q,n}[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = 0$. Assign $w_{u,n}[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = 1$ (or any other weight above 0).

If $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \not\mid\!\sim \neg(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$ and $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \mid\!\sim \neg(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$, then $(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$ will be at the same rank, q, as $(A_1\cdot\ldots\cdot A_{n-1})$, and $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$ will be at some lower rank u. Assign $w_{q,n}[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = w_{q,n-1}[A_1\cdot\ldots\cdot A_{n-1}]$, $w_{q,n}[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = 0$. Assign $w_{u,n}[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = 1$ (or any other weight above 0).

If $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \not\mid\!\sim \neg(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$ and $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \not\mid\!\sim \neg(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$, then $(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$ and $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$ are both at the same rank, q, as $A_1\cdot\ldots\cdot A_{n-1}$. Then assign $w_{q,n}[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = w_{q,n}[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = (1/2) \times w_{q,n-1}[A_1\cdot\ldots\cdot A_{n-1}]$ (or assign $w_{q,n}[A_1\cdot\ldots\cdot A_{n-1}\cdot C]$ and $w_{q,n}[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C]$ any weights above 0 you like, provided that $w_{q,n}[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] + w_{q,n}[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = w_{q,n}[A_1\cdot\ldots\cdot A_{n-1}]$).

If $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \mid\!\sim \neg(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$ and $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \mid\!\sim \neg(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$, then both $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$ and $(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$ will be at rank-ω, and $(A_1\cdot\ldots\cdot A_{n-1})$ must already be at rank-ω (no weights are assigned to rank-ω sentences).

Finally, for each sentence B above rank-ω:

if B has rank q, assign $w_{q,n}[B] =$
the sum of the values of $w_{q,n}[S]$ for all rank q state-descriptions S in S_n that logically entail B,

if B has some rank other than q, assign $w_{q,n}[B] = 0$.

Notice that if B is a sentence in language L_{n-1}, then B is logically equivalent to a disjunction of state-descriptions, each state-description having form $(A_1\cdot\ldots\cdot A_{n-1})$. It's easy to check that the above construction yields the following result:

if B has rank q, assign $w_{q,n}[B]$
= the sum of the values of $w_{q,n}[S]$ for all rank q state-descriptions S in S_n that logically entail B
= the sum of the values of $w_{q,n-1}[S]$ for all rank q state-descriptions S in S_{n-1} that logically entail B

if B has some rank other than q, assign $w_{q,n}[B] = 0 = w_{q,n-1}[B]$.

Now define $P_{|\sim}$ as follows:

$P_{n|\sim}[A \mid B] = 1$ when the rank of B is ω; and

$P_{n|\sim}[A \mid B] = w_{q,n}[A\cdot B]/w_{q,n}[B]$ when the rank of B is q (but not ω).

Clearly, when A and B are both in L_{n-1}:

if the rank of B is q (but not ω), then $P_{n|\sim}[A \mid B] = w_{q,n}[A\cdot B]/w_{q,n}[B] = w_{q,n-1}[A\cdot B]/w_{q,n-1}[B] = P_{n-1|\sim}[A \mid B]$;

if the rank of B is ω, then $P_{n|\sim}[A \mid B] = 1 = P_{n-1|\sim}[A \mid B]$.

It's easy to check that $P_{1|\sim}$ must be a *Popper function* on the language L_1 (which contains only one sentence letter). Suppose $P_{n-1|\sim}$ on a finite language L_{n-1} is a *Popper function* (i.e. satisfies the *Popper function* axioms). Then it is easy to show that $P_{n|\sim}$ is a *Popper function* that agrees with $P_{n-1|\sim}$ on L_{n-1}. (E.g. We've already shown that each truth-table probability function is a *Popper function* (Tm 15); it's easy to show that $P_{1|\sim}$ is a truth-table probability function; and whenever $P_{n-1|\sim}$ is a truth-table probability function – where each truth-table line is assigned the rank and weight that corresponds to the rank and weight for the state-description of L_{n-1} made true by that truth-table line – then $P_{n|\sim}$ as defined above must also be a truth-table probability function that extends $P_{n-1|\sim}$ to the language L_n.)

Thus, the sequence of functions $P_{n|\sim}$ on finite languages L_n is a nested sequence of *Popper functions*.

If the language L is finite, then it is the largest language L_n in the sequence: define $P_{|\sim}$ to be $P_{n|\sim}$.

If the language L is countably infinite, define $P_{|\sim}$ on language L as follows:

$P_{|\sim}[A \mid B] = P_{n|\sim}[A \mid B]$ for the first finite language L_n in the sequence that contains both A and B.

$P_{|\sim}$ satisfies the axioms for *Popper functions* on L (since any violation of a Popper function axiom by sentences of language L implies that very same violation for the same sentences on the first finite language L_n in the sequence that contains all those sentence).

Theorem 23: For each *Popper function* P, there is a *rational consequence relation* $|\!\sim$ that specifies the same ranks for sentences as P specifies. Furthermore, for each rank of $|\!\sim$, there are *additive* weighting functions w_q at each rank q above ω that specify the probability values for P as follows:

$P[A \mid B] = 1$ when the rank of B is ω; and

$P[A \mid B] = w_q[A\cdot B]/w_q[B]$ when the rank of B is q (but not ω).

The *additivity* of w_q means that whenever $\models \neg(B\cdot C)$ and $(B\lor C)$ has rank q, $w_q[B\lor C] = w_q[B] + w_q[C]$.

proof: Given *Popper function* P, let $|\!\sim$ be the consequence relation defined as follows:

$B\ |\!\sim A$ just when $P[A \mid B] = 1$.

Then $|\!\sim$ is a *rational consequence relation* (Tm 12), and $|\!\sim$ provides ranks to all sentences of the language *L* of P. We will show how to use the *Popper function* P to define a weighting function w_q on each rank q of $|\!\sim$. We then show that $|\!\sim$ together with these weighting functions yield back the *Popper function* P via the rules:

$P[A \mid B] = 1$ when the rank of B is ω for $|\!\sim$; and

$P[A \mid B] = w_q[A\cdot B]/w_q[B]$ when the rank of B is q (but not ω) for $|\!\sim$.

We will establish *additivity* along the way.

Here are the details.

The following lemma will prove useful.

(*) Important fact: Suppose $P[Y \mid Z] > 0$.

Then $P[X \mid (X\lor Y)\cdot Z]/P[Y \mid (X\lor Y)\cdot Z] = P[X \mid Z] / P[Y \mid Z]$.

So, when $X\lor Y \models Z$, $P[X \mid (X\lor Y)]/P[Y \mid (X\lor Y)] = P[X \mid Z] / P[Y \mid Z]$.

proof: $P[X \mid Z] = P[X\cdot(X\lor Y) \mid Z] = P[X \mid (X\lor Y)\cdot Z] \times P[X\lor Y \mid Z]$

$P[Y \mid Z] = P[Y\cdot(X\lor Y) \mid Z] = P[Y \mid (X\lor Y)\cdot Z] \times P[X\lor Y \mid Z]$

$P[X \mid Z] / P[Y \mid Z] = P[X \mid (X\lor Y)\cdot Z] / P[Y \mid (X\lor Y)\cdot Z]$. Done.

To specify the weighting functions, we first arrange the sentence letters of the language *L* for P in an "alphabetical order". Then, for each n, for the set of the first n sentence letters, let L_n be the language for sentential logic based on just those sentence letters. Construct the corresponding set of state-descriptions S_n for L_n, where each state-description in S_n has its sentences letters (or their negations) in alphabetical order. For set S_n of state-descriptions, find the rank of each of its members *S* according to $|\!\sim$, and assign it a weight $w_q(S)$ in the following way:

For each rank q, let the sentence F_q be the first state-description that has rank q (via the alphabetical ordering) among the sequence of sets of state-descriptions $\langle S_1, ..., S_n, ...\rangle$.

For a given set of state-descriptions S_n, each of its members takes the form $A_1\cdot...\cdot A_{n-1}\cdot A_n$, where each A_k is a sentence letter or its negation, and where A_n is either the "new" sentence letter, call it C, or its negation, $\neg C$, where C is a sentence letter that does not occur in language L_{n-1} (so doesn't occur in the state-descriptions in set S_{n-1}). Thus, $A_1\cdot...\cdot A_{n-1}\cdot A_n$ is of form $A_1\cdot...\cdot A_{n-1}\cdot C$ or $A_1\cdot...\cdot A_{n-1}\cdot\neg C$, where $A_1\cdot...\cdot A_{n-1}$ is a state description from set S_{n-1}. State-description $A_1\cdot...\cdot A_{n-1}$ is logically equivalent to $(A_1\cdot...\cdot A_{n-1}\cdot C)\lor(A_1\cdot...\cdot A_{n-1}\cdot\neg C)$. So the rank of $(A_1\cdot...\cdot A_{n-1}\cdot C)\lor(A_1\cdot...\cdot A_{n-1}\cdot\neg C)$ is the rank of $A_1\cdot...\cdot A_{n-1}$.

Let $A_1\cdot...\cdot A_{n-1}$ be a rank q state-description in S_{n-1}, and consider the state descriptions $A_1\cdot...\cdot A_{n-1}\cdot C$ and $A_1\cdot...\cdot A_{n-1}\cdot\neg C$ in S_n. Here is how we assign them weights:

(1) If $(A_1\cdot...\cdot A_{n-1}\cdot C)\lor(A_1\cdot...\cdot A_{n-1}\cdot\neg C)\ |\!\not\sim \neg(A_1\cdot...\cdot A_{n-1}\cdot C)$ (i.e. if $P[\neg(A_1\cdot...\cdot A_{n-1}\cdot C) \mid (A_1\cdot...\cdot A_{n-1}\cdot C)\lor(A_1\cdot...\cdot A_{n-1}\cdot\neg C)] < 1$) and $(A_1\cdot...\cdot A_{n-1}\cdot C)\lor(A_1\cdot...\cdot A_{n-1}\cdot\neg C)\ |\!\sim \neg(A_1\cdot...\cdot A_{n-1}\cdot\neg C)$ (i.e. if $P[\neg(A_1\cdot...\cdot A_{n-1}\cdot\neg C) \mid (A_1\cdot...\cdot A_{n-1}\cdot C)\lor(A_1\cdot...\cdot A_{n-1}\cdot\neg C)] = 1$),

then $(A_1\cdot...\cdot A_{n-1}\cdot C)$ is at the same rank as $(A_1\cdot...\cdot A_{n-1})$, and $(A_1\cdot...\cdot A_{n-1}\cdot\neg C)$is at some lower rank u.

Assign $w_q[A_1\cdot...\cdot A_{n-1}\cdot C] = P[A_1\cdot...\cdot A_{n-1}\cdot C \mid (A_1\cdot...\cdot A_{n-1}\cdot C)\lor F_q]/P[F_q \mid (A_1\cdot...\cdot A_{n-1}\cdot C)\lor F_q]$;

assign $w_v[A_1\cdot...\cdot A_{n-1}\cdot C] = 0$ for all ranks v other than q and ω;

assign $w_v[A_1\cdot...\cdot A_{n-1}\cdot\neg C] = 0$ for all ranks v other than u and ω;

assign $w_u[A_1\cdot...\cdot A_{n-1}\cdot\neg C] = P[A_1\cdot...\cdot A_{n-1}\cdot\neg C \mid (A_1\cdot...\cdot A_{n-1}\cdot\neg C)\lor F_u]/P[F_u \mid (A_1\cdot...\cdot A_{n-1}\cdot\neg C)\lor F_u]$, unless u is rank ω.

(2) If $(A_1\cdot...\cdot A_{n-1}\cdot C)\lor(A_1\cdot...\cdot A_{n-1}\cdot\neg C)\ |\!\sim \neg(A_1\cdot...\cdot A_{n-1}\cdot C)$ (i.e. if $P[\neg(A_1\cdot...\cdot A_{n-1}\cdot C) \mid (A_1\cdot...\cdot A_{n-1}\cdot C)\lor(A_1\cdot...\cdot A_{n-1}\cdot\neg C)] = 1$) and $(A_1\cdot...\cdot A_{n-1}\cdot C)\lor(A_1\cdot...\cdot A_{n-1}\cdot\neg C)\ |\!\not\sim \neg(A_1\cdot...\cdot A_{n-1}\cdot\neg C)$ (i.e. if $P[\neg(A_1\cdot...\cdot A_{n-1}\cdot\neg C) \mid (A_1\cdot...\cdot A_{n-1}\cdot C)\lor(A_1\cdot...\cdot A_{n-1}\cdot\neg C)] < 1$)

then $(A_1\cdot...\cdot A_{n-1}\cdot\neg C)$ is at the same rank as $(A_1\cdot...\cdot A_{n-1})$, and $(A_1\cdot...\cdot A_{n-1}\cdot C)$ is at some lower rank u.

Assign $w_q[A_1\cdot...\cdot A_{n-1}\cdot\neg C] = P[A_1\cdot...\cdot A_{n-1}\cdot\neg C \mid (A_1\cdot...\cdot A_{n-1}\cdot\neg C)\lor F_q]/P[F_q \mid (A_1\cdot...\cdot A_{n-1}\cdot\neg C)\lor F_q]$;

assign $w_v[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = 0$ for all ranks v other than q and ω;
assign $w_v[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = 0$ for all ranks v other than u and ω;
assign $w_u[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = P[A_1\cdot\ldots\cdot A_{n-1}\cdot C \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee F_u]/P[F_u \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee F_u]$, unless u is rank ω.

(3) If $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \mid\!\not\sim \neg(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$ (i.e. if $P[\neg(A_1\cdot\ldots\cdot A_{n-1}\cdot C) \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)] < 1$) and $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \mid\!\not\sim \neg(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$ (i.e. if $P[\neg(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)] < 1$),
then $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$ and $(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$ are both at the same rank as $A_1\cdot\ldots\cdot A_{n-1}$.
Assign $w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = P[A_1\cdot\ldots\cdot A_{n-1}\cdot C \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee F_q]$;
assign $w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = P[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)\vee F_q]$;
assign $w_v[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = 0$ for all ranks v other than q and ω;
assign $w_v[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = 0$ for all ranks v other than q and ω.

(4) If $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \mid\!\sim \neg(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$ and $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C) \mid\!\sim \neg(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$,
then both $(A_1\cdot\ldots\cdot A_{n-1}\cdot C)$ and $(A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)$ will be at rank ω, and $A_1\cdot\ldots\cdot A_{n-1}$ must also be at rank ω.
Assign $w_v[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = 0$ for all ranks v other than ω;
assign $w_v[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = 0$ for all ranks v other than ω.

The rest of the proof show that this way of assigning weightings yields the desired results.

This assignment of weights to state-descriptions yields the following when the rank of $A_1\cdot\ldots\cdot A_{n-1}$ is q (not ω):
$w_q[A_1\cdot\ldots\cdot A_{n-1}] = w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot C] + w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C]$.
To show this, we go through each of the cases above.

(1) Since $w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = 0$, we need only show that $w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = w_q[A_1\cdot\ldots\cdot A_{n-1}]$. Here is how.
In this case we have $P[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)\vee F_q] = 0$.
It follows from (*) (taking Z to be $(A_1\cdot\ldots\cdot A_{n-1})\vee F_q$, X to be $A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C$, and Y to be F_q), that (since $X\vee Y \models Z$) we have
$0 = P[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)\vee F_q] =$
$P[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]$.
It follows from (*) (taking Z to be $(A_1\cdot\ldots\cdot A_{n-1})\vee F_q$, X to be $A_1\cdot\ldots\cdot A_{n-1}\cdot C$, and Y to be F_q), that (since $X\vee Y \models Z$) we have
$w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = P[A_1\cdot\ldots\cdot A_{n-1}\cdot C \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee F_q] =$
$P[A_1\cdot\ldots\cdot A_{n-1}\cdot C \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q] =$
$P[A_1\cdot\ldots\cdot A_{n-1}\cdot C \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q] +$
$P[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]$ (since this term = 0) =
$P[A_1\cdot\ldots\cdot A_{n-1} \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q] = w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot C]$.

(2) Since $w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = 0$, we need only show that $w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = w_q[A_1\cdot\ldots\cdot A_{n-1}]$. Here is how.
Exactly like case (1), but with '$\neg C$' interchanged with 'C' throughout.

(3) We need to show that $w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] + w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = w_q[A_1\cdot\ldots\cdot A_{n-1}]$. Here is how.
It follows from (*) (taking Z to be $(A_1\cdot\ldots\cdot A_{n-1})\vee F_q$, X to be $A_1\cdot\ldots\cdot A_{n-1}\cdot C$, and Y to be F_q), that (since $X\vee Y \models Z$) we have
$w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot C] = P[A_1\cdot\ldots\cdot A_{n-1}\cdot C \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot C)\vee F_q] =$
$P[A_1\cdot\ldots\cdot A_{n-1}\cdot C \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]$.
It follows from (*) (taking Z to be $(A_1\cdot\ldots\cdot A_{n-1})\vee F_q$, X to be $A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C$, and Y to be F_q), that (since $X\vee Y \models Z$) we have
$w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = P[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C)\vee F_q] =$
$P[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]$.
Then $w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot C] + w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C] = P[A_1\cdot\ldots\cdot A_{n-1}\cdot C \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q] +$
$P[A_1\cdot\ldots\cdot A_{n-1}\cdot\neg C \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q] =$
$P[A_1\cdot\ldots\cdot A_{n-1} \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q]/P[F_q \mid (A_1\cdot\ldots\cdot A_{n-1})\vee F_q] = w_q[A_1\cdot\ldots\cdot A_{n-1}\cdot C]$.

Given this result, it is straightforward to show (by induction on m) that for each m, for all $n \leq m$, if S is a rank q state-description in $\mathcal{S}_n$, then $w_q[S]$ = the sum of the rank q state-description in $\mathcal{S}_m$ that logically entail S.

For each sentence B at a rank above rank ω, we assign B a weight at each rank as follows:

Given sentence B, let S_n be *the first* state-description set in the ordering that contains all of the sentences letter contained in B. Assign w_q[B] = the sum of the weights w_q[S] for all rank *q* state-descriptions S in S_n that logically entail B; assign w_q[B] = 0 if no rank *q* state-descriptions in S_n logically entails B.

This assignment provides a unique rank *q* weight to each sentence B, regardless of whether B itself is a rank *q* sentence. (Notice that when S is a rank *q* state-description that first occurs in S_n, this assignment yields w_u[S] = 0 when rank *u* is not the same rank as *q*; but when *q* it the rank of S, it makes the trivial assignment w_q[S] = w_q[S] – in which case the specification of w_q[S] above makes w_q[S] > 0. Also notice that any state-description that contains all of the sentence letters in B must either logically entail B or logically entail ¬B.)

Given sentence B, let S_m be *any* state-description set in the ordering that contains all of the sentences letter contained in B. It follows that w_q[B] = the sum of the weights of all rank *q* state-descriptions in S_m that logically entail B. To see why:

Let S_n be *the first* state-description set in the ordering that contains all of the sentences letter contained in B, and let S be any rank *q* state-description in S_n that logically entail B. Then, for any state-description set S_m with m > n, w_q[S] = the sum of the weights of all rank *q* state-descriptions in S_m that logically entail S – and since S logical entails B, each of these state-descriptions also logically entail B. Thus, given sentence B, let S_m be *any* state-description set in the ordering that contains all of the sentences letter contained in B. Then, w_q[B] = the sum of the weights w_q[S] for all rank *q* state-descriptions S in S_n that logically entail B = the sum of the weights of all rank *q* state-descriptions in S_m that logically entail the state-descriptions S in S_n that logically entail B = the sum of the weights of all rank *q* state-descriptions in S_m that logically entail B.

Additivity of w_q follows:

Suppose |= ¬(B·C) and (B∨C) has rank *q*. Then either (1) B has rank *q* and C has a rank below *q*, or (2) C has rank *q* and B has a rank below *q*, or (3) both B and C have rank *q*. Let L_m be the first language in the sequence that contains all sentence letters in B∨C. We consider the state-descriptions in S_m, the state-descriptions for a language L_m that contains all sentence letters in B and all sentence letters in C (although it may not be the first language to do so).

In case (1), no rank *q* state-description in S_n can logically entail C (otherwise C would have rank *q* or a higher rank), so w_q[C] = 0; and since |= ¬(B·C), no state-descriptions can entail both B and C; so, all and only the rank *q* state-descriptions in S_n that logically entail B also logically entail B∨C, so w_q[B∨C] = w_q[B]; thus, w_q[B∨C] = w_q[B] + w_q[C].

Case (2) is just like case (1), but with 'B' and 'C' exchanged.

In case (3), w_q[B] + w_q[C] = the sum of the weights of rank *q* state-descriptions that logically entail B and the weights of rank *q* state-descriptions that logically entail C = the sum of the weights of rank *q* state-descriptions that logically entail (B∨C) (since no state-description entails both B and C) = w_q[B∨C].

To complete the proof we only need to establish that P[A | B] = w_q[A·B]/w_q[B] when the rank of B is *q* (not ω) for |~. (For, obviously, whenever the rank of B is ω for |~, P[A | B] = 1: for, suppose the rank of B is ω for |~; then B |~ ¬B, so P[¬B | B] = 1, so P[B | B] + P[~B | B] = 2 ≠ 1, so P[A | B] = 1 since P is a *Popper function*.)

To see that this works, suppose B has rank *q*. We know that A·B cannot have rank above B, so its rank is *q* or lower.

1. If (A·B) has rank below *q* then P[A | B] = P[A·B | B] = 0; but in that case our construction clearly makes w_q[A·B] = 0 and w_q[B] > 0, so P[A | B] = w_q[A·B]/w_q[B].

2. Suppose A·B has rank *q* as well. Let S_n be the first state-description set that has all sentence letters of A·B, and let D be the disjunction of all rank *q* state-descriptions from S_n that logically entail B. Then P[A | B] = P[A·B | B] = P[A·B | D], and also w_q[D] = w_q[B]. Let E be the disjunction of all rank *q* state-descriptions from S_n that logically entail A·B. Then P[E | D] = P[A·B | D], and w_q[E] = w_q[A·B]. Thus, showing that P[E | D] = w_q[E]/w_q[D] will suffice, since P[A | B] = P[E | D] = w_q[E]/w_q[D] = w_q[A·B]/w_q[B].

Notice that E |= D (i.e. the state-descriptions that make up E are a subset of those that make up D. Without loss of generality we can take E to have the form S_1∨...∨S_k, and D has the form S_1∨...∨S_k∨S_{k+1}∨...∨S_m, where the S_j are the appropriate state-descriptions from S_n. *Additivity* yield $w_q[E] = \sum_{j=1}^{k} w_q[S_j]$ and $w_q[D] = \sum_{j=1}^{m} w_q[S_j]$.

For each state-description S_j, $w_q[S_j] = P[S_j \mid S_j \vee F_q] / P[F_q \mid S_j \vee F_q]$ (that's how their weights were defined). Further, abbreviating D as (S_1∨...∨S_m), from observation (*) we have (for S_j as X, F_q as Y, (S_1∨...∨S_m)∨F_q as Z, with X∨Y |= Z)

$P[S_j \mid S_j \vee F_q] / P[F_q \mid S_j \vee F_q] = P[S_j \mid (S_1 \vee ... \vee S_m) \vee F_q] / P[F_q \mid (S_1 \vee ... \vee S_m) \vee F_q]$. Thus,
$w_q[E] = \sum_{j=1}^{k} w_q[S_j] = \sum_{j=1}^{k} P[S_j \mid (S_1 \vee ... \vee S_m) \vee F_q] / P[F_q \mid (S_1 \vee ... \vee S_m) \vee F_q]$ and
$w_q[D] = \sum_{j=1}^{m} w_q[S_j] = \sum_{j=1}^{m} P[S_j \mid (S_1 \vee ... \vee S_m) \vee F_q] / P[F_q \mid (S_1 \vee ... \vee S_m) \vee F_q]$.
Then, since
$\sum_{j=1}^{m} P[S_j \mid (S_1 \vee ... \vee S_m) \vee F_q] = P[(S_1 \vee ... \vee S_m) \mid (S_1 \vee ... \vee S_m) \vee F_q]$ and
$\sum_{j=1}^{k} P[S_j \mid (S_1 \vee ... \vee S_m) \vee F_q] = P[(S_1 \vee ... \vee S_k) \mid (S_1 \vee ... \vee S_m) \vee F_q]$, we have

$$w_q[E] / w_q[D] = P[(S_1 \vee ... \vee S_k) \mid (S_1 \vee ... \vee S_m) \vee F_q] / P[(S_1 \vee ... \vee S_m) \mid (S_1 \vee ... \vee S_m) \vee F_q]$$
$$= P[(S_1 \vee ... \vee S_k) \mid (S_1 \vee ... \vee S_m)] / P[(S_1 \vee ... \vee S_m) \mid (S_1 \vee ... \vee S_m)]$$

(from (*) with X as $(S_1 \vee ... \vee S_k)$, Y as $(S_1 \vee ... \vee S_m)$, Z as $(S_1 \vee ... \vee S_m) \vee F_q$, with $X \vee Y \models Z$ – indeed, $X \models Y$ and $Y \models Z$, so $X \vee Y \models Z$ in this case).
So, $w_q[E] / w_q[D] = P[(S_1 \vee ... \vee S_k) \mid (S_1 \vee ... \vee S_m)] = P[E \mid D]$.

Therefore, $P[A \mid B] = P[E \mid D] = w_q[E]/w_q[D] = w_q[A \cdot B]/w_q[B]$.

Theorem 24: Each finite part of a *Popper function* P (the part of P defined on a finite sublanguage) is a *truth-table conditional probability function*.

proof: Follows from the ranked structure of *Popper functions*, Theorem 23, and the direct connection between state-descriptions and truth-table lines, as specified in the ranked truth-table construction described in the proof of Theorem 21.