The Semantic Theory of Truth, Falsehood and Conditionals

(RBA, 11-10-87)

A. Introduction

1) The Problem (some background).

a) My dissatisfaction with standard logic since 1947.

i) intuitive, (e.g., paradoxes of" material and strict implication", in deontic logic, Russell's paradox), ii) its failures to give answers to questions it should give answers to. (In Phi of Sci, contrary-to-fact conditionals, lawlike statements, confirmation,definitions of dispositional predicates, conditional probabilities).The problem:Standard Logic

b) My previous attempts:1962, "Subjunctive conditionals" -(A->-A), problems; 1968, "A formalistic approach to synonymity"; 1970's work on "containment" and "truth-operators";1986-7, different lines begin to converge.

c) Need a plausible, rigorous, semantic theory of truth to support both the theorems of standard logic and logical theorems and rules for the new conditionals.

2) The main intuitive concepts and ideas to be connected with symbolic expressions of this formal semantic system are:

a) 'p is false' is not the same as 'p is not true'
'p is true' is not synonymous with 'p is not false'

b) The theorems of logic are not universally true; rather, they are universally unfalsifiable.
Anti-theorems of logic (inconsistencies) are not universally false; rather, they are are universally incapable of being true, i.e., unverifiable.

c) truth and falsehood are always relative to a universe of discourse - a set of objects (the domain of the universe of discourse), and the properties and relationships which obtain among these objects.
(vs. Frege) truth and falsehood are relational properties, not objects.

d) The concept of an atomic statement's being true, presupposes 1) that all subject terms exist in the universe of discourse, and 2) the predicate applies to those terms in that universe of discourse.

e) Falsehood distinguished from non-truth: 'p is false' means 'the subject is in the universe of discourse, but the predicate does not apply to it'. 'p is not true' (or 'not(p is true))' means 'either p is false, or some of its subject terms are not in the domain of discourse.

f) relative to a given universe of discourse, any statement which refers only to entities not in the domain of that universe of discourse, is neither true nor false.

g) conditional statements are true if and only if the antecedent is true and the consequent is also; they are false if and only if the antecedent is true and the consequent is false. If the antecedent is not true then the conditional is neither true nor false.
(This is the 'Nicod conditional' or 'C-conditional')

B. A set-theoretic semantical theory of truth-conditions for the logical connectives '-' (for "it is not the case that..."), '&' (for "both...and---"), '<->' (for "...if and only if---"), '->' (for "if...then---") and '(Ex)' (for "for some x...") and all operators defined in terms of them. To capture the 'if...then---' (Nicod's conditional, not the truth-functional conditional) that I am trying to capture , I define four distinct predicates to consider in each case: 'is true', 'is false', 'is not true', and 'is not false'. The last three, we shall find, can be defined in terms of '-' and '...is true'.

1. Atomic wffs. Interpreted atomic wffs are true relative to a field of reference Rj, only if (i) the interpretations of all singular terms are members of the set DRj, and (ii) the n-tuple of these interpretations of its arguments is a member of the set-theoretical interpretation of its predicate:

a) I(Pi<t1,...,tn>)e{T"Rj}
    iff ({I(t1),...,I(tn)}cDRj & <I(t1),...,I(tn)>eI(Pi))

False sentences are sentences which conform to clause (i) but not to clause (ii):

b) I(Pi<t1,...,tn>)e{F"Rj}
    iff ({I(t1),...,I(tn)}cDRj & -(<I(t1),...,I(tn)>eI(Pi))

I(A) is not true with respect to Rj, if and only if it is not a member of the sentences which are true with respect to Rj. Thus defining NT('not true') and NF('not false') as follows,

I(A)e{NT"Rj} -df -(I(A)e{T"Rj}),
I(A)e{NF"Rj} -df -(I(A)e{F"Rj}),

thus atomic wffs belong to {NF"Rj} and {NT"Rj} as follows:

c) I(Pi<t1,...,tn>)e{NT"Rj}
    iff -({I(t1),...,I(tn)}cDRj & <I(t1),...,I(tn)>eI(Pi))

d) I(Pi<t1,...,tn>)e{NF"Rj}
    iff -({I(t1),...,I(tn)}cDRj & -(<I(t1),...,I(tn)>eI(Pi))

2. Molecular Wffs of Standard Logic (without quantifiers)

2.1 Negation:

a) I(-A)e{T"Rj} iff I(A)e{F"Rj}
b) I(-A)e{F"Rj} iff I(A)e{T"Rj}
c) I(-A)e{NT"Rj} iff -(I(A)e{T"Rj})
d) I(-A)e{NF"Rj} iff -(I(A)e{F"Rj})

2.2 Conjunction:

a) I(A.B)e{T"Rj} iff (I(A)e{T"Rj} & I(B)e{T"Rj})
b) I(A.B)e{F"Rj} iff ((I(A)e{F"Rj} v I(B)e{F"Rj})
    i.e.,((I(A)e{F"Rj} & I(B)e{F"Rj})
    v(-I(A)e{F"Rj} & I(B)e{F"Rj})
    v(I(A)e{F"Rj} & -I(B)e{F"Rj}))
c) I(A.B)e{NT"Rj} iff (-(I(A)e{T"Rj}) v -(I(B)e{T"Rj}))
d) I(A.B)e{NF"Rj} iff (-(I(A)e{F"Rj} v I(B)e{F"Rj})

2.3 The semantical rules for 'v', '->', and '<->' are derivable, using the usual definitions, from the above.

3. C-conditionals (without quantifiers)

A C-conditional (A->B) is true whenever the conjunction of A and B is true, and false whenever the truth-functional conditional (A->B) is false. But if the antecedent is false, unlike conjunction and the truth-functional conditional, it it is neither true nor false, in line with Nicod's account.

3.1 C-biconditionals:

a) I(A<->B)e{T"Rj} iff (I(A)e{T"Rj} & I(B)e{T"Rj})
b) I(A<->B)e{F"Rj} iff ((I(A)e{T"Rj} & I(B)e{F"Rj})
v(I(A)e{F"Rj} & (I(B)e{T"Rj}))
c) I(A<->B)e{NT"Rj} iff (-(I(A)e{T"Rj}) v -(I(B)e{T"Rj}))
d) I(A<->B)e{NF"Rj} iff (-(I(A)e{T"Rj} & I(B)e{F"Rj})
.-(I(A)e{F"Rj} & I(B)e{T"Rj}))

3.2 C-conditionals: [From 3.1 and '(A->B)'-df'(A<->(A.B))']

a) I(A->B)e{T"Rj} iff (I(A)e{T"Rj} & I(B)e{T"Rj})
b) I(A->B)e{F"Rj} iff (I(A)e{T"Rj} & I(B)e{F"Rj})
c) I(A->B)e{NT"Rj} iff (-(I(A)e{T"Rj})v -(I(B)e{T"Rj}))
d) I(A->B)e{NF"Rj} iff (-(I(A)e{T"Rj})v -(I(B)e{F"Rj}))

4. Quantified Sentences

4.1 "Existential" Quantification:

a) I((Ex)Øx)e{T"Rj} iff for some I'(Øx) in which I'(x)eDRj, but otherwise the same as in I(Øx), I'(Øx)e{T"Rj}.
b) I((Ex)Øx)e{F"Rj} iff for every I'(Øx) in which I'(x)eDRj, but otherwise the same as in I(Øx) and I'(Øx)e{F"Rj}.
c) I((Ex)Øx)e{NT"Rj} iff -(I(Ex)θx)e{T"Rj})
d) I((Ex)Øx)e{NF"Rj} iff -(I(Ex)Øx)e{F"Rj})

4.2 Universal Quantification:

[The semantical rules for '(∀x)' are derivable, using the usual definitions, from the above].

C. The Law of Trivalence

The Principle of Bivalence which is assumed in standard logic: says: "Of every proposition, P, one and only one of the following is true: 1) P is true or 2) P is false."

The Principle of Trivalence, provable in the semantics above says: "Of every indicative sentence, S, one and only one of the following is true: 1) S is true and not false,
or 2) S is false and not true,
or 3) S is neither true nor false."

It is fairly easy to see how this principle is proved by considering an atomic wff, Pa, and an intended field of reference, Rj. Let us put 'T(Pa)'for 'I(Pa)e{T"Rj}', 'F(Pa)'for 'I(Pa)e{F"Rj}', '-T(Pa)'for '-(I(Pa)e{T"Rj})' and '-F(Pa)'for '-(I(Pa)e{F"Rj})'. Then,

'T(Pa)' =df '(I(a)eDRj & <I(a)>e{I(P)})', i.e., (A & B)
'F(Pa)' =df '(I(a)eDRj & -<I(a)>e{I(P)})' i.e., (A & -B)
'-T(Pa)' =df '-(I(a)eDRj & <I(a)>e{I(P)})' i.e., -(A & B)
'-F(Pa)' =df '-(I(a)eDRj & -<I(a)>e{I(P)})' i.e., -(A & -B)

These four "truth-values" are not mutually exclusive; three pairs of them are mutually compatible, and three pairs are mutually incompatible, i.e., inconsistent. All other n-tuples of them are mutually incompatible, or inconsistent. Of the six possible pairs, the three mutally compatible pairs are:

T(Pa)&-F(Pa) which has the form ((A & B) & -(A & -B))
F(Pa)&-T(Pa) which has the form ((A & -B) & -(A & B))
-T(Pa)&-F(Pa) which has the form (-(A & B) & -(A & -B))

The three mutually incompatible pairs are:

T(Pa)&F(Pa) which has the form ((A & B) & (A & -B))
T(Pa)&-T(Pa) which has the form ((A & B) & -(A & B))
F(Pa)&-F(Pa) which has the form ((A & -B) & -(A & -B))

Any conjunction of 3 or more of these predicates will be inconsistent and thus not true of any referential field. The proof of the principle of trivalence follows these lines.

D. The truth-operator and Three-valued Truth-tables

To the primitive sentential operators of our logical language logic we have thus added, the letter 'T', called the "truth operator". Prefixed to any statement, S, in the object language (the language which refers only to objects, properties and relations in the intended field of reference), 'T(S)' is read "It is true that S". 'T(S)' is true if and only if "'S'is true" is.

Next, we adopt the following conventions:

'-T(A)&-F(A)' is replaceable by 'A has the truth-value 0'
'T(A)&-F(A)' is replaceable by 'A has the truth-value 1'
'F(A)&-T(A)' is replaceable by 'A has the truth-value 2'

We can now summarize the semantic theory above, so far as it relates to sentential logic, in the following truth-tables:

A	-A	&	0	1	2	v	0	1	2	->	0	1	2	<->	0	1	2	<=>	0	1	2	=>	0	1	2
0	00	0	0	0	2	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	21	1	0	1	2	1	1	1	1	1	0	1	2	1	0	1	2	1	0	1	2	1	0	1	2
2	12	2	2	2	2	2	0	1	2	2	1	1	1	2	0	2	1	2	0	2	0	2	0	0	0

'FA' ('it is false that A'), 'NTA' '(it is not true that A' and 'NFA'('it is not false that A') are defined from 'T' and '-':

FA syn(df) T-A) NTA syn(df) -TA) NFA syn(df) -T-A)

hence,

A	-A	TA	FA	NTA	NFA	(TA & -FA)	(FA & -TA)	(-TA & -FA)
0	00	20	20	1 0	1 0	2 2 1	2 2 1	1 1 1
1	21	11	21	2 1	1 1	1 1 1	2 2 2	2 2 1
2	12	22	12	1 2	2 2	2 2 2	1 1 1	1 2 2

E. Laws of Truth-Logic: Non-Contradiction, Excluded Middle.

Theorem-schemata of pure logic never have false instances (as opposed to "always have true instances"). The semantics above establishes that the '(-A v A)' is never false and the '(A & -A)' is never true ('UNF' means 'universally not false' i.e., never false; and 'UNT' means 'universally not true' or never true).

1) UNT(A&-A) [Basic postulate of non-contradiction]
2) UNF(-AvA) [First version of Excluded Middle]

In the extension of pure logic which contains the truth-operator, i.e., in Truth-logic, the Principle of Bivalence fails, but there remain three distinct Laws of Non-Contradiction, and three distinct Laws of Excluded Middle. That no instances of these laws will ever be false is shown by the three-valued truth-tables. These laws are:

3) UNF-(T(A)&F(A)) (contrariety of Truth and Falsehood)
4) UNF-(T(A)&-T(A)) \ (Laws of Non-Contradiction)
5) UNF-(F(A)&-F(A)) /
6) UNF(-T(A)vT(A)) \
7) UNF(-F(A)vF(A)) | (Laws of Excluded Middle)
8) UNF(-F(A)v-T(A))/

F. The standard rules for truth-functional sentential operators, as well as those for C-conditionals, are expressible using the T-operator and C-conditionals or C-biconditionals, and are proven unfalsifiable by the three-valued truth-tables.

(This is superior to standard logic because, '-TA' implies '(TA -> q)' in standard logic; but never implies '(TA -> q)')

'NOT'. The following rules can never lead to falsehood:
'A is false iff (-A) is True'; 'A is true iff (-A)' is false';
'If A is true, then A is not false';
'If A is false, then A is not true'.

UNF(F(A) <-> T(-A))) But not: UT(F(A) <-> T(-A))
UNF(T(A) <-> F(-A))) But not: UT(T(A) <-> F(-A))
UNF(T(A) -> NF(A))) But not: UT(T(A) -> NF(A))
But not the converse, (NF(NF(A) -> T(A))
UNF(F(A) -> NT(A)))
But not the converse, (NF(NT(A) -> F(A)))

'AND'. The following rules can never lead to falsehood:
'A is true and B is true if and only if (A&B) is true'
'(A&B) is false if and only if either A is false or B is false'

UNF((T(A) & T(B))<-> T(A & B))) UNF((F(A) v F(B))<->F(A & B))
UNF((NT(A)vNT(B))<->NT(A & B))) UNF((NF(A)& NF(B))<->NF(A & B))
UNF((T(A) & F(B)) -> F(A & B))) UNF((T(A) & NT(B))-> NT(A & B))
(but not converse) (but not converse)

'EITHER...OR'. The following rules can never yield falsehoods:
'(A or B) is true if and only if either A is true or B is true'.
'(A or B) is false if and only if A is false and B is false'.

UNF(T(A v B)<->(T(A) v T(B)))) UNF(F(A v B)<->(F(A) & F(B))))
UNF(NT(AvB)<->(NT(A)& NT(B)))) UNF(NF(AvB)<->(NF(A)v NF(B))))
UNF((T(A) & F(B)) -> T(A v B))) UNF((F(A) & NT(B))-> NT(A v B))
(but not converse) (but not converse)

THE C-CONDITIONAL:

The following rules will never yield a falsehood:
'('If A then B' is true if and only if A is true and B is true'
'('If A then B' is false if and only if A is true and B is not':

UNF(T(A =>B) <-> (TA & TB))) UNF(F(A =>B)<->(TA & NTB)))
UNF(NT(A =>B)<->(NTA v NTB))) UNF(NF(A =>B)<->(NTA v TB)))

Conjunction, the C-biconditional and C-conditionals:

UNF((A<->B) <-> ((A ->B) & (B ->A))) UNF(A ->B)<->(A<->(A&B)))

G. Remarks on the relation between this semantic theory of truth, and the proposed system of logic (i.e., "Analytic Logic")

T(A ⇒ B) ⇒ (TA·TB)
-T(A ⇒ B) ⇒ -(TA.TB) ⇒ (-TA v -TB)

1. Logical validity is not determined by the semantics of the truth predicate (as is said in standard logic); but must be consistent with it in the sense that no theorem-schema of logic can have a false instantiation, and no anti-theorem-schema can have a true one.

2. A valuable distinction can be draw between 'tautology' and 'validity', and between 'inconsistency' and 'contra validity', with respect to truth and falsity. We define a tautology as the negation of an inconsistency (in standard Logic the theorems are tautologies, the anti-theorems are inconsistencies). Tautologies can never be false and inconsistencies can never be true. In the logic of C-conditionals it is possible for a wff to be both inconsistent and tautologous, and thus logically incapable of having either a true or a false instantiation. (E.g., ((A.-A)=>A)). Valid wffs are now defined as wffs which are both tautologous and consistent; i.e., wffs with some true instantiations but no false ones. Contra-valid wffs have some false instantiations, but no true ones.

3. Given this definition of "valid" we can hold that an inference is valid iff the C-conditional with an antecedent formed by conjoining the premisses and with the conclusion as consequent is valid. It avoids the 'paradoxical' rule that every proposition follows validly from any inconsistent set of propositions. Other constraints, (on tautologous conditionals) avoid the principle that every tautology follows from any proposition. This definition of 'valid inference' or 'logical consequence' takes the condition, "it is never the case that all premisses are true and the conclusion is not" as a necessary condition of valid inference, but not as a sufficient condition (as is done in standard logic).

4. The theorems and anti-theorems of logic, though compatible with truth-logic, can be developed without it. That is, the essential predicates of logic (e.g., 'is inconsistent', 'is tautologous', 'is valid') can be semantically defined without reference to any semantic theory of truth. The schemata of logic have as constants only sentential operators, ('and','or','not','if...then ', etc.) and quantifiers. Logic can be developed formally as a logic of predicates which, by themselves are never either true or false, but can stand in semantic (meaning) relationships of one being synonymous or not, of one being contained in the other or not, of one being opposed to (inconsistent with) another or not). These are sufficient to give a semantic theory to support logic. The relation of inconsistency and tautologies to truth and falsity, arises when one goes beyond meanings and relations of meanings, to relations of meanings to non-linguistic fields of reference.

APPENDIX - Three-valued Truth Table Tests

```
Basic postulate of non-contradiction
  1) UNT(A & -A))     UNT(A & -A))      But not: UF(A & -A))
       1  0 0 00        1 (0 0 00)                 2(  0     )
       1  1 2 21        1 (1 2 21)                 1(  2     )
       1  2 2 12        1 (2 2 12)                 1(  2     )
Basic version of Excluded Middle
  2)  UNF(-A v A))      UNF(-A v A))      But not:  UT(-A v A))
        1  00 0 0         1 (00 0 0)                  2(    0   )
        1  21 1 1         1 (21 1 1)                  1(    1   )
        1  12 1 2         1 (12 1 2)                  1(    1   )
Three Laws of Non-Contradition:
  3) Contrariety of Truth and Falsehood but not Bivalence:
      UT-(T(A) & F(A)))    or,     UF(T(A) & F(A)))
       11(2 0   2 2 0 )            1 (2 0   2 2 0 )
       11(1 1   2 2 1 )            1 (1 1   2 2 1 )
       11(2 2   2 1 2 )            1 (2 2   2 1 2 )
                                    Satisfiability of
      Failure of Bivalence          Denial of Bivalence
        UT(T(A) v F(A))             (-T(A) & -F(A))
         2(2 0   2 2 0 )             12 0   1 12 0 )
         1(1 1   1 2 1 )             21 1   2 12 1 )
         1(2 2   1 1 2 )             12 2   2 21 2 )
     Non-contradictoriness of        Satisfiability of
     'Not true and not false'
        (NT(A) & NF(A))             -(NT(A) & NF(A))
        ( 1(0) 1  1(0))             2( 1(0) 1  1(0))
        ( 2(1) 2  1(1))             1( 2(1) 2  1(1))
        ( 1(2) 2  2(2))             1( 1(2) 2  2(2))

  4) UT-(T(A) & NT(A))     or,      UF(T(A) & NT(A))
     1 1(2 0  2  1 0 )              1 (2 0  2  1 0 )
     1 1(1 1  2  2 1 )              1 (1 1  2  2 1 )
     1 1(2 2  2  1 2 )              1 (2 2  2  1 2 )

  5) UT-(F(A) & NF(A))     or,      UF(F(A) & NF(A))
     1 1(2(0) 2  1(0))              1 (2(0) 2  1(0))
     1 1(2(1) 2  1(1))              1 (2(1) 2  1(1))
     1 1(1(2) 2  2(2))              1 (1(2) 2  2(2))

Three Laws of Excluded Middle
  6) UT(NT(A) v T(A))     or,     UF(-NT(A) & -T(A))
     1 ( 1(0) 1 2(0))             1 (21 (0) 2 12(0))
     1 ( 2(1) 1 1(1))             1 (12 (1) 2 21(1))
     1 ( 1(2) 1 2(2))             1 (21 (2) 2 12(2))
  7) UT(NF(A) v F(A))     or,     UF(-NF(A) & -F(A))
     1 ( 1(0) 1 2(0))             1 (21 (0) 2 12(0))
     1 ( 1(1) 1 2(1))             1 (21 (1) 2 12(1))
     1 ( 2(2) 1 1(2))             1 (12 (2) 2 21(2))
  8) UT(NT(A) v NF(A))    or,     UF(-NT(A) & -NF(A))
     1 ( 1(0) 1  1(0))            1 (21 (0) 2 21 (0))
     1 ( 2(1) 1  1(1))            1 (12 (1) 2 21 (1))
     1 ( 1(2) 1  2(2))            1 (21 (2) 2 12 (2))
 Some wffs which are logically neither true nor false:
  ((A & -A) -> A)  (-A ->(A ->A))  ((A ->-A) <-> -A)
```

```
0 0 00  0  0     00 0  0 0 0      0 0 00   0  00
1 2 20  0  1     21 0  1 1 1      1 2 21   0  21
2 2 10  0  2     12 0  2 0 2      2 0 12   0  12
```

<u>D. The standard truth-functional rules for sentential operators, as well as those for C-conditionals:</u>

<u>'NOT'</u>. The following rules can never lead to falsehood:
'A is false iff -A is True'; 'Ais true iff -A' is false';
'If A is true, then A is not false';
'If A is false, then A is not true'.

```
 9) UNF(F(A) <-> T(-A)))        But not: UT(F(A) <-> T(-A))
      1  2 0   0  2 00                    2         0
      1  2 1   0  2 21                    2         0
      1  1 2   1  1 12                    1         1

10) UNF(T(A) <-> F(-A)))        But not: UT(T(A) <-> F(-A))
      1  2 0   0  2 00                    2         0
      1  1 1   1  1 21                    1         1
      1  2 2   0  2 12                    2         0

11) UNF(T(A) -> NF(A)))         But not: UT(T(A) -> NF(A))
      1  2 0  0  1  0                     2        0
      1  1 1  1  1  1                     1        1
      1  2 2  0  2  2                     2        0
    But not the converse
      (NF(A) -> T(A))
        1 0  2  2 0
12) UNF(F(A) -> NT(A)))        UT(NF(F(A) -> NT(A)))
      1  2 0  0  1  0           1  1       0
      1  2 1  0  2  1           1  1       0
      1  1 2  1  1  2           1  1       1
     But not the converse
       (NT(A) -> F(A))
         1 0  2  2 0
-|A \
0|0  |  means:    "If V(A)=0 then V(-A)=0
2|1  |           or if V(A)=1 then V(-A)=2
1|2 /            or if V(A)=2 then V(-A)=1."

(NT(A) & NF(A)) -> (NT(-A) & NF(-A))  I.e., (V(A)=0 <-> V(-A)=0)
 12 0  1 12 0   1   12 00  1 12 00
 21 1  2 12 1   0   12 21  2 21 21
 12 2  2 21 2   0   21 12  2 12 12
 (T(A) & NF(A)) -> (F(-A) & NT(-A))   I.e., (V(A)=1 <-> V(-A)=2)
  2 0  2 12 0   0   2 00  2 12 00
  1 1  1 12 1   1   1 21  1 12 21
  2 2  2 21 2   0   2 12  2 21 12
 (F(A) & NT(A)) -> (T(-A) & NF(-A))   I.e., (V(A)=2 <-> V(-A)=1)
  2 0  2 12 0   0   2 00  2 12 00     [Note: with '->' each of
  2 1  2 21 1   0   2 21  2 21 21      these become universally
  1 2  1 12 2   1   1 12  1 12 12      true, UT, and UNF]
```

<u>'AND'</u>. The following rules can never lead to falsehood:
'A is true and B is true if and only if (A&B) is true'

'(A&B) is false if and only if either A is false or B is false'

UNF((T(A) & T(B))<-> T(A & B)))										
1	2	0	2	2	0	0	2	0	0	0
1	1	1	2	2	0	0	2	1	0	0
1	2	2	2	2	0	0	2	2	2	0
1	2	0	2	1	1	0	2	0	0	1
1	1	1	1	1	1	1	1	1	1	1
1	2	2	2	1	1	0	2	2	2	1
1	2	0	2	2	2	0	2	0	2	2
1	1	1	2	2	2	0	2	1	2	2
1	2	2	2	2	2	0	2	2	2	2

UNF((F(A) v F(B))<->F(A & B))										
1	2	0	2	2	0	0	2	0	0	0
1	2	1	2	2	0	0	2	1	0	0
1	1	2	1	2	0	**1**	1	2	2	0
1	2	0	2	2	1	0	2	0	0	1
1	2	1	2	2	1	0	2	1	1	1
1	1	2	1	2	1	**1**	1	2	2	1
1	2	0	1	1	2	**1**	1	0	2	2
1	2	1	1	1	2	**1**	1	1	2	2
1	1	2	1	1	2	**1**	1	2	2	2

UNF((NT(A)vNT(B))<->NT(A & B)))										
1	1	0	1	1	0	**1**	12	0	0	0
1	2	1	1	1	0	**1**	12	1	0	0
1	1	2	1	1	0	**1**	12	2	2	0
1	1	0	1	2	1	**1**	12	0	0	1
1	2	1	2	2	1	0	21	1	1	1
1	1	2	1	2	1	**1**	12	2	2	1
1	1	0	1	1	2	**1**	12	0	2	2
1	2	1	1	1	2	**1**	12	1	2	2
1	1	2	1	1	2	**1**	12	2	2	2

UNF((NF(A)& NF(B))<->NF(A & B))										
1	1	0	1	1	0	1	1	0	0	0
1	1	1	1	1	0	1	1	1	0	0
1	2	2	2	1	0	0	2	2	2	0
1	1	0	1	1	1	1	1	0	0	1
1	1	1	1	1	1	1	1	1	1	1
1	2	2	2	1	1	0	2	2	2	1
1	1	0	2	2	2	0	2	0	2	2
1	1	1	2	2	2	0	2	1	2	2
1	2	2	2	2	2	0	2	2	2	2

UNF((T(A) & F(B)) -> F(A & B)))										
1	2	0	2	2	0	0	2	0	0	0
1	1	1	2	2	0	0	2	1	0	0
1	2	2	2	2	0	0	1	2	2	0
1	2	0	2	2	1	0	2	0	0	1
1	1	1	2	2	1	0	2	1	1	1
1	2	2	2	2	1	0	1	2	2	1
1	2	0	2	1	2	0	1	0	2	2
1	1	1	1	1	2	**1**	1	1	2	2
1	2	2	2	1	2	0	1	2	2	2

(not converse)

UNF((T(A) & NT(B))-> NT(A & B))										
1	2	0	2	1	0	0	1	0	0	0
1	1	1	1	1	0	**1**	1	1	0	0
1	2	2	2	1	0	0	1	2	2	0
1	2	0	2	2	1	0	1	0	0	1
1	1	1	2	2	1	0	2	1	1	1
1	2	2	2	2	1	0	1	2	2	1
1	2	0	2	1	2	0	1	0	2	2
1	1	1	1	1	2	**1**	1	1	2	2
1	2	2	2	1	2	0	1	2	2	2

(not converse)

<u>Some other Corollaries provably UNF by truth-tables:</u>

UNF((T(A) & T(B)) -> T(A&B)))
UNF((T(A) & F(B)) -> F(A&B))) (not converse)
UNF((F(A) & T(B)) -> F(A&B))) (not converse)
UNF((F(A) & F(B)) -> F(A&B))) (not converse)

(UNF((V(A)-0 & V(B)-0) -> V(A&B)-0) v
UNF((V(A)-1 & V(B)-0) -> V(A&B)-0) v
UNF((V(A)-2 & V(B)-0) -> V(A&B)-2) v
UNF((V(A)-0 & V(B)-1) -> V(A&B)-0) v
UNF((V(A)-1 & V(B)-1) <-> V(A&B)-1) v
UNF((V(A)-2 & V(B)-1) -> V(A&B)-2) v
UNF((V(A)-0 & V(B)-2) -> V(A&B)-2) v
UNF((V(A)-1 & V(B)-2) -> V(A&B)-2) v
UNF((V(A)-2 & V(B)-2) -> V(A&B)-2))

The meaning of:

&	0	1	2
0	0	0	2
1	0	1	2
2	2	2	2

<u>'EITHER...OR'</u>. The following rules can never yield falsehoods: '(A or B) is true if and only if either A is true or B is true'.

'(A or B) is false if and only if A is false and B is false'.

```
UNF(T(A v B)<->(T(A) v T(B))))
 1  2 0 0 0  0  2 0  2 2 0
 1  1 1 1 0  1  1 1  1 2 0
 1  2 2 0 0  0  2 2  2 2 0
 1  1 0 1 1  1  2 0  1 1 1
 1  1 1 1 1  1  1 1  1 1 1
 1  1 2 1 1  1  2 2  1 1 1
 1  2 0 0 2  0  2 0  2 2 2
 1  1 1 1 2  1  1 1  1 2 2
 1  2 2 2 2  0  2 2  2 2 2
    ^                ^
```

```
UNF(F(A v B)<->(F(A) & F(B))))
 1  2 0 0 0  0  2 0  2 2 0
 1  2 1 1 0  0  2 1  2 2 0
 1  2 2 0 0  0  1 2  2 2 0
 1  2 0 1 1  0  2 0  2 2 1
 1  2 1 1 1  0  2 1  2 2 1
 1  2 2 1 1  0  1 2  2 2 1
 1  2 0 0 2  0  2 0  2 1 2
 1  2 1 1 2  0  2 1  2 1 2
 1  1 2 2 2  1  1 2  1 1 2
    ^                ^
```

```
UNF(NT(AvB)<->(NT(A)& NT(B))))
 1  1  000  1  1  0 1 1  0
 1  2  110  0  2  1 2 1  0
 1  1  200  1  1  2 1 1  0
 1  2  011  0  1  0 2 2  1
 1  2  111  0  2  1 2 2  1
 1  2  211  0  1  2 2 2  1
 1  1  002  1  1  0 1 1  2
 1  2  112  0  2  1 2 1  2
 1  1  222  1  1  2 1 1  2
    ^             ^
```

```
UNF(NF(AvB)<->(NF(A)v NF(B))))
 1  1  000  1  1  0 1 1  0
 1  1  110  1  1  1 1 1  0
 1  1  200  1  2  2 1 1  0
 1  1  011  1  1  0 1 1  1
 1  1  111  1  1  1 1 1  1
 1  1  211  1  2  2 1 1  1
 1  1  002  1  1  0 1 2  2
 1  1  112  1  1  1 1 2  2
 1  2  222  0  2  2 2 2  2
    ^               ^
```

```
UNF((T(A) & F(B))-> NF(A v B)))
 1   2 0  2 2 0  0  12 0 0 0
 1   1 1  2 2 0  0  12 1 1 0
 1   2 2  2 2 0  0  12 2 0 0
 1   2 0  2 2 1  0  12 0 1 1
 1   1 1  2 2 1  0  12 1 1 1
 1   2 2  2 2 1  0  12 2 1 1
 1   2 0  2 1 2  0  12 0 0 2
 1   1 1  1 1 2  1  12 1 1 2
 1   2 2  2 1 2  0  21 2 2 2
          (not converse)
```

```
UNF((F(A) & NT(B))-> NT(A v B))
 1   2 0  2 1  0  0  12 0 0 0
 1   2 1  2 1  0  0  21 1 1 0
 1   1 2  1 1  0  1  12 2 0 0
 1   2 0  2 2  1  0  21 0 1 1
 1   2 1  2 2  1  0  21 1 1 1
 1   1 2  2 2  1  0  21 2 1 1
 1   2 0  2 1  2  0  12 0 0 2
 1   2 1  2 1  2  0  21 1 1 2
 1   1 2  1 1  2  1  12 2 2 2
          (not converse)
```

<u>Some other Corollaries provably UNF by truth-tables:</u>

```
UNF((F(A) & T(B)) -> T(AvB)))     (not converse)
UNF((T(A) & T(B)) -> T(AvB)))     (not converse)
UNF((T(A) & F(B)) -> T(AvB)))     (not converse)
UNF((F(A) & F(B)) -> F(AvB)))
```

```
(UNF((V(A)-0 & V(B)-0) -> V(AvB)-0) v  \    The
 UNF((V(A)-1 & V(B)-0) -> V(AvB)-1) v   |   meaning
 UNF((V(A)-2 & V(B)-0) -> V(AvB)-0) v   |   of:
 UNF((V(A)-0 & V(B)-1) -> V(AvB)-1) v   |   v|0 1 2
 UNF((V(A)-1 & V(B)-1) -> V(AvB)-1) v    >  0|0 1 0
 UNF((V(A)-2 & V(B)-1) -> V(AvB)-1) v   |   1|1 1 1
 UNF((V(A)-0 & V(B)-2) -> V(AvB)-0) v   |   2|0 1 2
 UNF((V(A)-1 & V(B)-2) -> V(AvB)-1) v   |
 UNF((V(A)-2 & V(B)-2) -> V(AvB)-2) )  /
```

<u>THE C-CONDITIONAL:</u>

The following rules will never yield a falsehood:

'('If A then B' is true if and only if A is true and B is true')
'('If A then B'is false if and only if A is true and B is false')

UNF(	T(	A	->	B)	<->	(TA	&	TB)))
1	2	0	0	0	0	20	2	20
1	2	1	0	0	0	11	2	20
1	2	2	0	0	0	22	2	20
1	2	0	0	1	0	20	2	11
1	1	1	1	1	**1**	11	1	11
1	2	2	0	1	0	22	2	11
1	2	0	0	2	0	20	2	22
1	2	1	2	2	0	11	2	22
1	2	2	0	2	0	22	2	22
	^						^	

UNF(	F(	A	->	B)	<->	(TA	&	FB)))
1	2	0	0	0	0	20	2	20
1	2	1	0	0	0	11	2	20
1	2	2	0	0	0	22	2	20
1	2	0	0	1	0	20	2	21
1	2	1	1	1	0	11	2	21
1	2	2	0	1	0	22	2	21
1	2	0	0	2	0	20	2	12
1	1	1	2	2	**1**	11	1	12
1	2	2	0	2	0	22	2	12
	^						^	

UNF(	NT(	A	->	B)	<->	(N	TA	v	N	TB)))
1	1	0	0	0	**1**	1	0	1	1	0
1	1	1	0	0	**1**	2	1	1	1	0
1	1	2	0	0	**1**	1	2	1	1	0
1	1	0	0	1	**1**	1	0	1	2	1
1	2	1	1	1	0	2	1	2	2	1
1	1	2	0	1	**1**	1	2	1	2	1
1	1	0	0	2	**1**	1	0	1	1	2
1	1	1	2	2	**1**	2	1	1	1	2
1	1	2	0	2	**1**	1	2	1	1	2
	^							^		

UNF(	NF(	A	->	B)	<->	(N	TA	v	NFB)))
1	1	0	0	0	**1**	1	0	1	120
1	1	1	0	0	**1**	2	1	1	120
1	1	2	0	0	**1**	1	2	1	120
1	1	0	0	1	**1**	1	0	1	121
1	1	1	1	1	**1**	2	1	1	121
1	1	2	0	1	**1**	1	2	1	121
1	1	0	0	2	**1**	1	0	1	212
1	2	1	2	2	0	2	1	2	212
1	1	2	0	2	**1**	1	2	1	212
	^							^	

C-BICONDITIONALS AND C-CONDITIONALS

C-biconditional and conjunction of C-conditionals:

UNF((	A	<->	B)	<->	((A	->	B)	&	(B	->	A)))
1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	0	1
1	2	0	0	0	2	0	0	0	0	0	2
1	0	0	1	0	0	0	1	0	1	0	0
1	1	1	1	**1**	1	1	1	1	1	1	1
1	2	2	1	0	2	0	1	2	1	2	2
1	0	0	2	0	0	0	2	0	2	0	0
1	1	2	2	0	1	2	2	2	2	0	1
1	2	0	2	0	2	0	2	0	2	0	2
		^						^			

UNF((	A	->	B)	<->	(A	<->	(A&B)))
1	0	0	0	0	0	0	000
1	1	0	0	0	1	0	100
1	2	0	0	0	2	0	220
1	0	0	1	0	0	0	001
1	1	1	1	**1**	1	1	111
1	2	0	1	0	2	0	221
1	0	0	2	0	0	0	022
1	1	2	2	0	1	2	122
1	2	0	2	0	2	0	222
		^				^	

<u>RE TARSKI</u>: Tarski's 'T(A) <-> A', and analogues, are not VALID: though UNF.

UNF(<u>T(A) <-> A</u>)	**UT**(TA <-> A)	**UNF**(T(A) <-> A)	**UNF**(A <-> NF(A))
1 2 0 0 0	**2** 0	1 (2 0 0 0)	1 0 0 1 0
1 1 1 **1** 1	1 1	1 (1 1 **1** 1)	1 1 **1** 1 1
1 2 2 **1** 2	1 1	1 (2 2 0 2)	1 2 0 2 2
		^not sim ^	^not sim ^

But, '(TA -> A)' and 'T(TA -> A)' are Valid:

(TA<->(TA&A)) hence	(T(A)->A), and	(TT(A)<->T(A)),	T(TA -> A)
(20 0 2 00)	2 0 0 0	22 0 0 2 0)	1 20 1 0
(11 **1** 1 11)	1 1 **1** 1	11 1 **1** 1 1	1 11 1 1
(22 0 2 22)	2 2 0 2	22 2 0 2 2	1 22 1 2
^ sim ^		^--sim--^	

(Note that in these cases, though the truth-table has no F's and at least one '1', the two sides do not have the same analytical truth-table, thus though 'TA<->A' has no false instantiations, and can be satified, the two expressions 'TA','NFA' and 'A' are not synonymous, thus not logically equivalent in analytic logic. But also the referential meaning of 'T(A)' differs from the referential meaning of 'A'; the first describes a relation between a sentence and its meaning and a field of reference; the second does not refer to sentences or meanings).

The concept of a field of reference is the concept of a set of entities to which the property of being inconsistent can not apply; inconsistency can only be a property of linguistic expressions. To say a sentence is true of a certain field of reference is to say that its content corresponds to what is in the field of reference. Since the concept of being inconsistent can never correspond any entity, property or relation in a field of reference,

1)	**UNF**(TTA	<->	TA))
	1	220	0	20
	1	111	1	11
	1	222	0	22
2)	**UNF**(FFA	<->	NFA)
	1	120	0	1 0
	1	121	1	1 1
	1	212	0	2 2

But not: **UNF**(FFA	<->	TA)
	1	2	2
	1	1	1
	2	0	2