~P	=	Not-P
(P · Q)	=	Both P and Q
(P ∨ Q)	=	Either P or Q
(P ⊃ Q)	=	If P then Q
(P ≡ Q)	=	P if and only if Q

Propositional logic

1. Any capital letter is a wff.
2. The result of prefixing any wff with "~" is a wff.
3. The result of joining any two wffs by "•" or "∨" or "⊃" or "≡" and enclosing the result in parentheses is a wff.

Two translation rules

Put "(" wherever you see "both," "either," or "if."

Either not A or B = $(\sim A \vee B)$

Not either A or B = $\sim(A \vee B)$

Group together parts on either side of a comma.

If A, then B and C = $(A \supset (B \cdot C))$

If A then B, and C = $((A \supset B) \cdot C)$

$\sim P$	=	Not-P
$(P \cdot Q)$	=	Both P and Q
$(P \vee Q)$	=	Either P or Q
$(P \supset Q)$	=	If P then Q
$(P \equiv Q)$	=	P if and only if Q

Propositional translations

Put "(" wherever you see "both," "either," or "if."

Group together parts on either side of a comma.

“I went to Paris *and* I went to Quebec.”

P	Q	(P • Q)
0	0	0
0	1	0
1	0	0
1	1	1

(0 • 0) = 0
(0 • 1) = 0
(1 • 0) = 0
(1 • 1) = 1

“(P • Q)” claims that *both* parts are true.
“(P • Q)” is a *conjunction*; P and Q are its *conjuncts*.

“I went to Paris *or* I went to Quebec.”

P	Q	$(P \vee Q)$
0	0	0
0	1	1
1	0	1
1	1	1

$(0 \vee 0) = 0$
$(0 \vee 1) = 1$
$(1 \vee 0) = 1$
$(1 \vee 1) = 1$

“$(P \vee Q)$” claims that *at least one* part is true.
“$(P \vee Q)$” is a *disjunction*; P and Q are its *disjuncts*.

“*If* I went to Paris, *then* I went to Quebec.”

P	Q	$(P \supset Q)$
0	0	1
0	1	1
1	0	0
1	1	1

$(0 \supset 0) = 1$
$(0 \supset 1) = 1$
$(1 \supset 0) = 0$
$(1 \supset 1) = 1$

“$(P \supset Q)$” says we *don’t* have the first part true and the second false.
“$(P \supset Q)$” is a *conditional*; P is the *antecedent* and Q the *consequent*.

Falsity implies anything.	$(0 \supset \quad) = 1$
Anything implies truth.	$(\quad \supset 1) = 1$
Truth doesn’t imply falsity.	$(1 \supset 0) = 0$

P	Q	(P ≡ Q)
0	0	1
0	1	0
1	0	0
1	1	1

(0 ≡ 0) = 1
(0 ≡ 1) = 0
(1 ≡ 0) = 0
(1 ≡ 1) = 1

"I went to Paris, *if and only if* I went to Quebec."

"(P ≡ Q)" claims that both parts have the *same* truth value.
"(P ≡ Q)" is a *biconditional*.

P	~P
0	1
1	0

~0 = 1
~1 = 0

"I *didn't* go to Paris."

"~P" has the *opposite* value of "P."
"~P" is a *negation*.

Basic Truth Equivalences

AND	OR	IF-THEN	IFF	NOT
$(0 \cdot 0) = 0$	$(0 \vee 0) = 0$	$(0 \supset 0) = 1$	$(0 \equiv 0) = 1$	
$(0 \cdot 1) = 0$	$(0 \vee 1) = 1$	$(0 \supset 1) = 1$	$(0 \equiv 1) = 0$	$\sim 0 = 1$
$(1 \cdot 0) = 0$	$(1 \vee 0) = 1$	$(1 \supset 0) = 0$	$(1 \equiv 0) = 0$	$\sim 1 = 0$
$(1 \cdot 1) = 1$	$(1 \vee 1) = 1$	$(1 \supset 1) = 1$	$(1 \equiv 1) = 1$	
both parts are true	*at least one* part is true	we don't have *first true & second false*	both parts have *same* truth value	*reverse* the truth value

Falsity implies anything.
Anything implies truth.
Truth doesn't imply falsity.

Truth Evaluations

Assume that A=1 and X=0.

What is the truth value of "∼(A · X)"?

- Plug in "1" and "0" for the letters.
- Simplify from the inside out, until you get "1" or "0."

$$\sim(A \cdot X)$$
$$\sim(1 \cdot 0)$$
$$\sim 0$$
$$1$$

Unknown Evaluations

Assume that T=1 and F=0 and U=? (unknown).

What is the truth value of "(U · F)"?

"(? · 0)" has to be 0 because:

An AND with one part 0 is 0.	It comes out 0 both ways: (1 · 0) = 0 and (0 · 0) = 0.

Complex Truth Tables: do one for “(P ≡ ~Q)”

P	Q	(P ≡ ~Q)
0	0	0
0	1	1
1	0	1
1	1	0

(0 ≡ ~0) = (0 ≡ 1) = 0
(0 ≡ ~1) = (0 ≡ 0) = 1
(1 ≡ ~0) = (1 ≡ 1) = 1
(1 ≡ ~1) = (1 ≡ 0) = 0

- Write the formula: “(P ≡ ~Q)”
- On the left, write each letter in the formula: “P” and “Q”
- Below this, write each truth combination; n letters give 2^n truth combinations.
- Figure out the value of the formula on each combination.

VALID = No possible case has premises all true and conclusion false.	~~1 1 ∴ 0~~

1 1 ∴ 0	INVALID = Some possible case has premises all true and conclusion false.

Truth-table test: Construct a truth table showing the truth value of the premises and conclusion for all possible cases. The argument is **VALID** if and only if no possible case has premises all true and conclusion false.

C	D	$(C \supset D)$,	D	∴	C	
0	0	1	0		0	
0	1	1	1		0	← **Invalid**
1	0	0	0		1	
1	1	1	1		1	

On the short-cut truth-table test, evaluate easier wffs first and cross out rows that couldn't have true premises and a false conclusion.

C D	(C ⊃ D),	D	∴	C
0 0	~~– – – – – – – –~~	~~0~~	~~– – – –~~	~~– – – –~~
0 1		1		
1 0	~~– – – – – – – –~~	~~0~~	~~– – – –~~	~~– – – –~~
1 1		1		

First do "D" – and cross out rows that couldn't be 110.

C	D	$(C \supset D)$,	D	∴	C
~~0~~	~~0~~		~~0~~		
0	1		1		0
~~1~~	~~0~~		~~0~~		
~~1~~	~~1~~		~~1~~		~~1~~

Then do "C" – and cross out rows that couldn't be 110.

C	D	$(C \supset D)$,	D	∴	C
~~0~~	~~0~~		~~0~~		
0	1	1	1		0
~~1~~	~~0~~		~~0~~		
~~1~~	~~1~~		~~1~~		~~1~~

Finally, do "$(C \supset D)$."

← **Invalid**

VALID = No possible case has premises all true and conclusion false.	~~1~~ ~~1~~ ~~∴ 0~~

1 1 ∴ 0	INVALID = Some possible case has premises all true and conclusion false.

Truth-assignment test: Set each premise to 1 and the conclusion to 0. Figure out the truth value of as many letters as possible. The argument is **VALID** if and only if no possible way to assign 1 and 0 to the letters will keep the premises all 1 and conclusion 0.

$$\begin{array}{l} (L \vee R) = 1 \\ \sim L = 1 \\ \therefore\ R = 0 \end{array} \quad \rightarrow \quad \begin{array}{l} (L^0 \vee R^0) = 1 \\ \sim L^0 = 1 \\ \therefore\ R^0 = 0 \end{array} \quad \rightarrow \quad \begin{array}{l} \underline{\quad 0 \quad} \text{ Valid} \\ (L^0 \vee R^0) \neq 1 \\ \sim L^0 = 1 \\ \therefore\ R^0 = 0 \end{array}$$

Harder Translations

Translate "but" ("yet," "however," "although," and so on) as "and."	Northwestern played, *but* it won = $(N \cdot W)$
Translate "unless" as "or."	You'll die *unless* you give me your money = $(D \vee M)$
Translate "just if" and "iff" (a logician word) as "if and only if."	I'll take the job *just if* you pay me a million = $(J \equiv M)$

The part after "if" ("provided that," "assuming that," and so on) is the if-part (the antecedent, the part before the horseshoe).

Gensler is happy *if* Michigan wins.
= *If* Michigan wins, Gensler is happy.
= $(M \supset H)$

The part after "only if" is the then-part (the consequent, the part after the horseshoe). (Or write "⊃" for "only if.")

You're alive *only if* you have oxygen.
= *Only if* you have oxygen are you alive.
= $(A \supset O)$ = $({\sim}O \supset {\sim}A)$

Oxygen is *sufficient for* life = $(O \supset L)$

Oxygen is *necessary for* life = $(\sim O \supset \sim L)$

Oxygen is *necessary and sufficient for* life = $(O \equiv L)$

Harder Translations

- BUT = YET = HOWEVER = ALTHOUGH = AND.
- UNLESS = OR.
- JUST IF = IFF = IF AND ONLY IF.

- The part after “if” (“provided that,” “assuming that,” and so on) goes *before* the horseshoe.
- The part after “only if” goes *after* the horseshoe.

- Oxygen is *sufficient for* life = $(O \supset L)$
- Oxygen is *necessary for* life = $(\sim O \supset \sim L)$
- Oxygen is *necessary and sufficient for* life = $(O \equiv L)$

Idiomatic arguments

1. Identify the conclusion.

These often indicate premises:	∴	*These often indicate conclusions:*
Because, for, since, after all … I assume that, as we know … For these reasons …		Hence, thus, so, therefore … It must be, it can't be … This proves (or shows) that …

2. Translate into logic, using wffs. Add implicit premises if needed.

3. Test for validity.

S-Rules

AND

	This and that.	$\dfrac{(P \cdot Q)}{P,\ Q}$	AND statement, so both parts are true.
∴	This.		
∴	That.		

NOR

	Not either this or that.	$\dfrac{\sim(P \vee Q)}{\sim P,\ \sim Q}$	NOT-EITHER is true, so both are false.
∴	Not this.		
∴	Not that.		

NIF

	False if-then.	$\dfrac{\sim(P \supset Q)}{P,\ \sim Q}$	FALSE IF-THEN, so first part true, second part false.
∴	First part true.		
∴	Second part false.		

S-Rules: When can we simplify?

	CAN SIMPLIFY	CAN'T
AND	$(P \cdot Q) \rightarrow P, Q$	$\sim(P \cdot Q)$
NOR	$\sim(P \vee Q) \rightarrow \sim P, \sim Q$	$(P \vee Q)$
NIF	$\sim(P \supset Q) \rightarrow P, \sim Q$	$(P \supset Q)$

Recall our basic truth tables:

For an AND to be true, both parts have to be true.
For an OR to be false, both parts have to be false.
For an IF-THEN to be false, we need part 1 true and part 2 false.

I-Rules

Not both are true. This one is true. ∴ The other isn't.	$\sim(P \cdot Q)$ P ——— $\sim Q$	$\sim(P \cdot Q)$ Q ——— $\sim P$	*Deny* AND. *Affirm* one part. ∴ *Deny* other part.
At least one is true. This one isn't. ∴ The other is.	$(P \vee Q)$ $\sim P$ ——— Q	$(P \vee Q)$ $\sim Q$ ——— P	*Affirm* OR. *Deny* one part. ∴ *Affirm* other part.

IF-THEN. Affirm first. ∴ Affirm second.	$(P \supset Q)$ P ——— Q	IF-THEN. Deny second. ∴ Deny first.	$(P \supset Q)$ $\sim Q$ ——— $\sim P$

I-Rules: When can we infer?

With NAND or OR, premises must *alternate* affirming and denying.

Deny AND. *Affirm* one part. ∴ *Deny* other part.	NAND	$\sim(P \cdot Q),\ P\ \rightarrow\ \sim Q$ $\sim(P \cdot Q),\ Q\ \rightarrow\ \sim P$
Affirm OR. *Deny* one part. ∴ *Affirm* other part.	OR	$(P \vee Q),\ \sim P\ \rightarrow\ Q$ $(P \vee Q),\ \sim Q\ \rightarrow\ P$

With IF-THEN, we need part 1 true or part 2 false: "(+ ⊃ -)."

$(P \supset Q),\ P\ \rightarrow\ Q$

$(P \supset Q),\ \sim Q\ \rightarrow\ \sim P$

S- and I-Rules

S-rules

$(P \cdot Q) \rightarrow P, Q$

$\sim(P \vee Q) \rightarrow \sim P, \sim Q$

$\sim(P \supset Q) \rightarrow P, \sim Q$

I-rules

$\sim(P \cdot Q), P \rightarrow \sim Q$

$\sim(P \cdot Q), Q \rightarrow \sim P$

$(P \vee Q), \sim P \rightarrow Q$

$(P \vee Q), \sim Q \rightarrow P$

$(P \supset Q), P \rightarrow Q$

$(P \supset Q), \sim Q \rightarrow \sim P$

Extended S- and I-rule inferences

$$\frac{\sim((A \cdot B) \supset \sim C)}{(A \cdot B),\ C}$$

FALSE IF-THEN,
so first part true,
second part false.

$$\frac{\sim(\$\$\$ \supset \#\#\#)}{\$\$\$,\ \sim\#\#\#}$$

$$\frac{((A \cdot B) \supset \sim C) \qquad \sim(A \cdot B)}{\text{nil}}$$

IF-THEN:
need first part true
or second part false

$$\frac{(\$\$\$ \supset \#\#\#) \qquad \sim\$\$\$}{\text{nil}}$$

Logic gates and computers

A	B	(A • B)
0	0	0
0	1	0
1	0	0
1	1	1

An AND-GATE gives an output voltage if and only if both inputs have an input voltage.