Contents

Chapter 1

The Beginning

Sep. 3, 2025

1.1 Russell's Paradox

. . . or, when things (in MATH) go "sideways" . . .

A *motivational* tale.

1.1.1 Example. (Briefly about *set notation*)

Important. You do not need to know *ANY* set theory to understand "Russell's Paradox".

This is just a short tale of **misuse of Mathematics**.

Sets are *collections* of MATH objects.

We represent sets either by explicit *listing*, like this:

- $\{0\}$
- $\{\$, \#, 3, 42\}$
- $\{0, 1, 2, 3, 4, \ldots\}$

or by some "*defining property*": *The set of all* x[a] *that make* $P(x)$ *true*, in symbols

$$S = \{x : P(x)\} \tag{1}$$

[a]Strictly speaking you *DON'T* collect the various shapes and colours of the letter x. There is only ONE x. The expression "set of all x such that $P(x)$ is true" is sloppy for "the set of all VALUES of x such that $P(x)$ is true".

As we know from discrete MATHs, (1) says the *same thing* as the statement

$$x \in S \equiv P(x) \tag{2}$$

read "for any value of x, $x \in S$ is *equivalent* to $P(x)$"

Why so? Because $P(x)$ is an *entrance condition*!

A *value* of x is placed in the set S IF and *ONLY* IF (iff) said value *passes the test* $P(x)$.

Wait! Shouldn't I have written (2) as

$$x \in S \equiv P(x) \text{ is true} \tag{2'}$$

Nope. When mathematicians state $P(x)$ for some unspecified *fixed* (value of) x they mean "$P(x)$ is true" for that value.

Cantor believed (good grief! "believed"?! as did the mathematician and philosopher Frege) that, for any property $P(x)$, (1) defines a set —they never proved this; just "*believed*" and used it (!!) in Cantorian Set Theory.

Which is neither here nor there because they never said what a set is. They allowed ANY collection of (mathematical) objects to be a set!

They said: Check the meaning of the word "set" in a *dictionary*!

Russell begged to differ, so he said: "Oh, yeah? How about this? Is it a '*set*'?"

$$R = \{x : \overbrace{x \notin x}^{A\ Property\ of\ x}\ \}$$

where the "entry condition"/property "$P(x)$" here is SPECIFICALLY

$$x \notin x$$

Now, by (2) we have

$$x \in R \equiv x \notin x$$

If R *IS* a set, then we can plug it IN the set variable x above to obtain THE CONTRADICTION

$$R \in R \equiv R \notin R$$

How do we avoid Russell's *contradiction*?

By *admitting* that R is *NOT* a set so we do not allow the substitution! □

MEMO to Cantor/Frege: "Collections/sets/aggregates, etc. see dictionary(!)" do NOT all have the same "rights"!

Some "collections" like "R" above **cannot** be substituted into a set variable! Because they are NOT sets!

How can we separate the two?

1. **Use axioms**

OR

2. Use the Russell idea: **Sets don't just happen!** They are built **by stages!**

1.2 Enters Formal Logic!

How do we get out out of this contradiction and many other contradictions? —**yes, there were many others in Cantor's Set Theory work.**

- Cantor never said what sets really are and how they are built. He just used a *dictionary of synonyms* instead of a definition! (collection, class, aggregate, etc., he suggested as dictionary synonyms.)
- MUST use logic (he did not) to argue how sets behave and what they are.

WE, however in 1090A, will never deal with sets EVER again. So *we leave the task to set theorists* ([Jec78, Lev79, Tou03b]) or to EECS 1028 (or EECS 1019).

We only felt we should motivate the case for Logic at the expense of Cantor's approach to Set Theory.

So Cantor was sloppy about what a set IS

OR **how sets get formed**.

We also do know that Euclid did not define what **ARE** the "**points**, **lines**, or **planes**"; **yet his geometry is free of contradiction**.

> Not just there is none found, BUT IN FACT we have a **PROOF** that none will be **EVER** found, because they do NOT exist!

How come? He used the **Axiomatic Method and Logic**.

Sep. 8, 2025

Formal Logic —*means SYNTACTICALLY PERFORMED; Based on FORM.*

Just like Programming! FORM is of Utmost Importance! You do not/Cannot "handwave" to a computer!!

Logic was invented by Russell and Whitehead, and especially **Hilbert**, to *salvage* Mathematics from "antinomies" and "paradoxes", both words derived from Greek, and both meaning **contradictions**.

And remember. Only ONE contradiction is enough to destroy a theory!
Destroy how?
Well, the contradictory theory "thinks" that EVERY formula (statement) is a theorem. So it is USELESS.

How does formal logic *salvage* Mathematics?

Once you learn your axioms and your rules you will hardly ever write faulty proofs.

You cannot pull **fake facts** *off the air* —**any more than you can pull fake programming instructions off the air!!!**— but your facts *MUST* be *axioms* or *PREVIOUSLY proved* theorems

At the same time, the *rules* of logic that you use

MUST *NOT* DEPEND *on an* arbitrary choice (yours, Cantor's, or mine).

Connection of Formal Logic with Programming

(1) In programming we use *syntactic rules* to write a *program* in order to enable the computer to solve some problem computationally.

(2) In logic you use the syntactic rules to write a *proof* that establishes a **theorem** —colloquially, that is a "true statement of MATH"— so that the whole process (**writing the proof and communicating it**) convinces another math-literate person that the proof is "correct".

Kinds of logic reasoning that we will thoroughly examine and use in this course.

1. Equational logic —also known as *calculational logic*.

   Introduced by [DS90] and simplified by [GS94] and later by [Tou08] to make it accessible to undergraduates. Software Engineers use it.

2. Hilbert-style logic. This is the logic which most people use to write their mathematical arguments in publications, lectures, etc.

Logic is meant to certify mathematical truths syntactically.

Logic is normally learnt by

- A thorough study and effort to **memorise Definitions, Rules and Axioms.**

- *A LOT* of practice.

- By presenting and teaching it *gradually*, namely

  1. **First, learning the *Propositional Logic* (also known as *Boolean Logic*).**

     Here one learns how logical statements *combine* using *connectives* familiar from programming like OR, AND, and NOT.

     Boolean logic is *not expressive enough* to formulate statements about mathematical objects. Naturally, if you cannot ask it —

a question about such objects— *then you get no answer either*.

2. **Next, learning *Predicate Logic* (also known as *First-Order Logic*).**

   This is the full logic of the **mathematician, the software engineer and computer scientist** as it lets you formulate and explore statements that are about *mathematical objects* like numbers, strings and trees, programs (small, large and VERY large) and many others.

The axioms and rules of the first logic above are $PART$ of the second. So it is like learning a PART of a programming language —say PYTHON— first and then turning to learning and mastering the ENTIRE language.

1.3 A REVIEW of "strings"

1.3.1 Definition. (Strings; also called Expressions)

1. What is a *string* *over* *some* alphabet *of symbols*?

   It is an **ordered finite sequence** of symbols from the alphabet —with **no gaps or other "separators" between symbols**.

   1.3.2 Example. If the alphabet is $\{a, b\}$ then here are a few strings:

   (a) a

   (b) $aaabb$

   (c) $bbaaa$

   (d) $bbbbbbb$

□

What do we mean by "ordered"? We mean that *order, LENGTH AND multiplicity matter*!

For example, $aaabb$ and $bbaaa$ are different strings.

So are $aabbb$ and $baaaa$.

We indicate this by writing $aaabb \neq bbaaa$ and also $aabbb \neq baaaa$.

Moreover, $aa \neq aaaa$.

Two strings are equal iff [†] they have the same length n and at *each* position —from 1 to n— both strings have the same symbol. So, $aba = aba$, but $aa \neq a$ and $aba \neq baa$.

[†] If and only if.

1.3.1 A Bad Alphabet

Consider the alphabet $B = \{a, aa\}$.

This is bad. WHY?

Because if we write the string **aaa** over this alphabet we do not know what we mean by just **looking** at the string!

Do we mean THREE a like

$$a \quad a \quad a$$

Or do we mean

$$a \quad aa$$

Or perhaps

$$aa \quad a$$

We say that the alphabet B leads to *ambiguity*.

Since we use NO separators —like a space or a comma— between symbols in denoting strings we MUST ALWAYS choose alphabets with **single-symbol** *items.*

2. *Names* of strings: A, A'', A_5, B, C, S, T.

   What for? CONVENIENCE AND EASE OF EXPRESSION.

   Thus $A = bba$ gives the string bba the name A.

   Names vs *IS*: "Practicing" mathematicians and computer scientists take a sloppy attitude towards using the verb "BE/IS".

   When they say "let A be a string" they mean "let A name a string".

Same as in "let x be a rational number". Well x is not a number at all!

It is a letter!

We mean "let x **STAND** for, or **NAME**, a rational number"

3. *Operations on strings*: *Concatenation*. From strings aab and baa, concatenation in the order given yields the string $aabbaa$.

If A is a string (meaning names a string) and B is another, then their concatenation AB is not a concatenation of the names but *is a concatenation of the contents*. If $A = aaaa$ and $B = 101$ then $AB = aaaa101$.

Incidentally,

$$BA = 101aaaa \neq aaaa101 = AB$$

Thus *in* general *concatenation is* not commutative *as we say*.

Why "in general"?

Well, if $X = aa$ and $Y = a$ then $XY = aaa = YX$.
Special cases where concatenation does commute exist!

4. *Associativity* of concatenation.

   It is expressed as $(AB)C = A(BC)$ where bracketing here denotes **invisible** METAsymbols (*they are NOT part of any* string!) that simply INDICATE the order in which we GLUE, from left to right.

   At the left of the "=" we first concatenate A and B and then "*glue*" C at the *right* end.

   $$A \quad B \quad C$$

   if $A = 1, B = 2, C = 3$ then $A(BC) = 123$
   NOT $1(23)$

   To the right of "=" we first glue B and C and then glue A to the *left* of the result.

   In either case we did not change the relative positions of A, B and C.

   The property is self-evident.

   I can now skip most brackets and write $(ABC)D$ and you know what I mean! I can also just write $ABCD$ and I mean the same string.

5. *Empty* string. A string with *no symbols*, hence with *length* 0. Denoted by λ.

$$\lambda Q = Q, \ and \ Q\lambda = Q$$

How is λ different than $\emptyset$ the empty *set*?

Well one is of *string* *type* and the other is of *set type*. So? The former is an ORDERED empty set, the latter is an UNORDERED empty set that moreover is *oblivious to repetitions*.

I mean, $aaa \neq a$ but $\{a, a, a\} = \{a\}$.

6. Clearly, for any string A we have $A\lambda = \lambda A = A$ as concatenation of λ adds nothing to either end.

7. *Substrings*. A string A is a *substring* of B iff A appears *as is* as a *part* of B.

 So if $A = aa$ and $B = aba$ then A is *NOT* a substring of B. Its members both appear in B (the two a) but are *not* together as they are in A. *A does not appear "as is"*.

So if $A = aa$ and $B = baab$ then A *IS* a substring of B. 

Can we get rid of all this bla-bla with a proper definition? Sure:

1.3.3 Definition. *A is a substring of B iff for some strings (named) Q and R we have $B = QAR$.* □

 We also say A is *part* of B. 

8. *Prefix* and *suffix*. A is a *prefix* of B if for some string V, $B = AV$.

So A is part of B up in front!

A is a *suffix* of B if for some string U, $B = UA$. □

Example: λ is a prefix and a suffix, indeed a part, of *any* string B. Here are the "proofs" of the two cases I enumerated:

- $B = \lambda B$
- $B = B\lambda$

WHAT ABOUT THE THIRD CASE?

Well, $B = B\lambda\lambda$.

Sep. 10, 2025

1.4 The Boolean well-formed-formulas (wff)

The Syntax of logic.

Boolean Logic at first!

Boolean logic is the "Algebra of statements". We start with *atomic* statements and build complex statements using "glue" as I call the Boolean *connectives*

$$\neg, \wedge, \vee, \rightarrow, \equiv$$

Atomic statements have NO glue! We usually denote them by p, q, r with or without primes or subscripts.

E.g., p, q, r, r'''_{105} all are (meaning, all stand for) *unspecified* atomic statements.

Boolean logic has precisely two *specific* statements ("constants"). Read on!

Examples of statements that Boolean logic can express:

p, $(\neg p)$ and also $((p \vee q) \wedge r)$. And more!

Can I *see inside* atomic statements like p to see what they *mean*?

NO!! We cannot!

But we can assign *arbitrarily* "true" or "false" *values* to atomic statements and then proceed to see how these *truth values* **propagate** when I *apply glue*.

That is *all* that Boolean logic can do.

And this ends up being quite useful! Read on!

1.4.1 Definition. (Alphabet of Boolean Symbols)

A1. *Names* for *variables*, which we call "propositional" or "Boolean" variables.

These are p, q, r, with or without primes or subscripts (indices) (e.g., $p, q, r, p', q_{13}, r'''_{51}$ are all names for Boolean variables).

Thus we have INFINITELY MANY Boolean variables. **This will be useful later!**

A2. Two *symbols* denote the Boolean *constants*, $\top$ and $\bot$. We pronounce them "top" and "bot" respectively.

What are $\top$ and $\bot$ good for? We will soon see!

A3. (Round) brackets, i.e., "(" and ")" (employed without the quotes, of course).

A4. Boolean "**connectives**" that I will usually call "*glue*".

We use glue to put a formula together much like we do so when we build model cars or airplanes or houses.

The symbols for Boolean connectives (*glue*) are

$$\neg \ \wedge \ \vee \ \rightarrow \equiv \tag{1}$$

and are read from left to right as "negation, conjunction, disjunction, implication, equivalence". □

We stick to the above symbols —EXACTLY as written!!!— for glue (no pun!) in this course! *Just as we do in programming,*

NO DEVIATION IS PERMITTED!!!

You *cannot use* any symbols you **please** or "**like**". Can you do so in **C++, JAVA, or Pascal? Of course, NOT!**

SPEAKING BY ANALOGY, You use *THE* symbols of A Programming Language as *THEY ARE GIVEN*.

If not, your program does NOT work and your GRADE bottoms!

Same holds in logic!

1.5 Building Formulas.

1.5.1 Definition. (Formula Construction (process)) A *formula construction* (in the text also called "*formula calculation*"†) is any *finite (ordered) sequence of strings over the alphabet* of Boolean logic* $\mathcal{V}$ that obeys the following three specifications:

C0. We write **ONE** string per line (vertically).

C1. At any step we may write precisely one symbol from *categories* **A1.** or **A2.** above (1.4.1), that is, variables $(p, q'', r'''_{33}$, etc.) or constants ($\bot$ or $\top$).

C2. At any step we may write precisely *one* **string** *of the form* $(\neg A)$, *as long as* we have written the string (*named*) A already at a previous step.

So, "$(\neg A)$" is a string that has "$(\neg$" (no quotes) as a prefix, then it has a part we named A, and then it has "$)$" (no quotes) as a suffix.

I must stress that the letter A names the string that was written down earlier.

> In the construction, we did not write "A" but rather wrote the *string*-**contents** of "A".

†We can also say "Formula Building".

*"Over the Alphabet": Using exclusively symbols from the Alphabet $\mathcal{V}$ that we adopted.

> Just as in a *program*: When you issue the command "**print** X" you mean to print what the X contains as value —what it names. You do *not* mean to print the **letter** "X"!

C3. At any step we may write precisely *ONE* **of the strings** $(A \wedge B)$, $(A \vee B)$, $(A \to B)$, $(A \equiv B)$, *as long as* we have already written *EACH* of the TWO strings A and B earlier. □

We do not care which we wrote first, A or B.

1.5.2 Definition. (Boolean formulas (wff)) Any string A over the alphabet $\mathcal{V}$ (**A1.**–**A4.**) is called *a Boolean formula* or *a propositional formula* —in short **wff**— *iff* A is a string that appears in some formula construction. □

Let's parse the "**iff**" that is ubiquitous in Definitions like this one:

iff 1. **The "IF" part:** **IF** the string (that we NAMED) A **appears in a formula construction**, then it is a **wff**.

iff 2. **The "ONLY IF" part:** I have got a **wff** (NAMED) B in my pocket. Well, I am guaranteed that there is a formula construction out there where (a copy of) B appears!

1.5.3 Remark. That is:

Every string that appears in a formula construction is a wff. **The definition also says**,

Conversely, "*do you want to know if A is a wff?* ***Just make sure you can build a formula construction*** *where A appears.*" □

1.5.4 Example. We write formula constructions **vertically**, per definition. Below I *also* use numbering and annotation (in "$\langle \ldots \rangle$" brackets) to explain each step.

-

(1)	$\bot$	$\langle$const$\rangle$
(2)	p	$\langle$var$\rangle$
(3)	$(\neg\bot)$	$\langle(1) + \neg\rangle$
(4)	$\bot$	$\langle$const$\rangle$
(5)	$\top$	$\langle$const$\rangle$

Note that we *can* have *redundancy* and *repetitions*.

Ostensibly the only nontrivial info in the above is that $(\neg\bot)$ is a formula. **But it also establishes that $\bot$ and $\top$ and p are formulas. How so?**

- (1) $\bot$ $\langle$const$\rangle$
  (2) p $\langle$var$\rangle$
  (3) $(\neg\top)$ $\langle$oops!$\rangle$
  (4) $\bot$ $\langle$const$\rangle$
  (5) $\top$ $\langle$const$\rangle$

  Why the "oops"? The above is wrong at step (3). I have not written $\top$ in the construction *as I should before* I attempted to use it!

□

Chapter 2

Properties of the wff

Here we **speak** about wff —and discover some useful properties they have— before we get to our main task, eventually, of USING wff *in proofs*.

2.1 Boolean Wff

Let us repeat

2.1.1 Definition. (Boolean formulas, or wff) A *string* (or *expression*) A over the alphabet of Boolean symbols $\mathcal{V}$ is called a *Boolean formula* or a *Boolean well-formed formula* (in short *wff*) ***iff*** it occurs in *some* formula construction.

The set of all wff we denote by the all-capitals **WFF**.

The wff that are either propositional variables

$$p, q, p'', r_{123}, \ldots$$

or $\bot$ or $\top$, in short, *glue-less* , we call *Atomic* wff.

□

Notation. META names. We often want to say things such as "... bla-bla ... *for all variables* p ...".

▶ Well this is not exactly right! **There is only ONE variable p!**

We get around this difficulty by having *informal names* (in the *metatheory* as we say) for Boolean variables: $\mathbf{p}, \mathbf{q}, \mathbf{r}'$, etc.

Any such bold face informal variable can stand for *any* actual variable of our alphabet $\mathcal{V}$ whatsoever.

So "all variables $\mathbf{p}$" means "any of the actual variables $p, q, r_{1110001}, \ldots$ that $\mathbf{p}$ may stand for" while "all p" is *meaningless*!

We can give a definition of formulas that is *independent* from formula constructions: OK, the above Definition 1.5.2 says that A is a wff iff it appears in a construction as

1. Atomic: $\bot, \top, \mathbf{p}$
2. A *negation* $(\neg B)$, where B $\overbrace{\text{appeared earlier in the construction}}^{so\ is\ a\ wff}$
3. A string $(B \wedge C)$ or $(B \vee C)$ or $(B \to C)$ or $(B \equiv C)$, where B *and* C $\overbrace{\text{appeared earlier in the construction}}^{so\ are\ wffs}$

BUT we can say "B (so did C) Appeared EARLIER" differently:

"B (same for C) is a wff"

So,

2.1.2 Definition. (The Recursive Definition of wff!!!) An expression A over $\mathcal{V}$ is a wff just in case A is ONE of:

(1) Atomic $(\mathbf{p}, \bot, \top)$

OR

(2) $(\neg B)$, $(B \wedge C)$, $(B \vee C)$, $(B \rightarrow C)$, $(B \equiv C)$, *where B and C are wff.* □

2.1.3 Remark. The formulas $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \equiv B)$ are pronounced in English, from left to right, "not A", "A and B", "A or B", "if A then B" (but also " A implies B"), "A is equivalent to B". □

Sep. 15, 2025

2.1.4 Example. In the formula below "$\vee$" is the *last* glue applied.

$$((A \to B) \quad \underset{\text{last glue}}{\vee} \quad C)$$

□

2.1.5 Remark. The wff in Remark 2.1.3 have the same names as their "last glue", namely, *negation, conjunction, disjunction, implication* and *equivalence*. □

Pause. Why did I say "LAST" glue?

2.1.6 Example. Using 1.5.2 let us verify that $((p \vee q) \vee r)$ is a wff.
Well, here is a formula construction written with annotations:

(1)	p	$\langle$atomic$\rangle$
(2)	q	$\langle$atomic$\rangle$
(3)	r	$\langle$atomic$\rangle$
(4)	$(p \vee q)$	$\langle 1 + 2 + \vee$-glue$\rangle$
(5)	$((p \vee q) \vee r)$	$\langle 4 + 3 + \vee$-glue$\rangle$

Do we have to write down *all* the atomic wff at the very *beginning*? *Not really*, but it is important to write them BEFORE they are *used* in the construction!

So, this works too:

(1)	p	$\langle$atomic$\rangle$
(2)	q	$\langle$atomic$\rangle$
(3)	$(p \vee q)$	$\langle 1 + 2 + \vee$-glue$\rangle$
(4)	r	$\langle$atomic$\rangle$
(5)	$((p \vee q) \vee r)$	$\langle 4 + 3 + \vee$-glue$\rangle$

Recursively:

$$\overbrace{\underbrace{\underbrace{(}_{atomic}\overbrace{p}^{atomic} \vee \overbrace{q}^{atomic}\underbrace{)}_{wff} \vee \overbrace{r}^{atomic}}_{wff}}^{wff}$$

□

Intuitively, immediate predecessors of a wff are the formulas on which we applied the last glue.

2.1.7 Definition. (Immediate predecessors (i.p.))

1. No atomic formula has immediate predecessors.

2. Any of the following wff
   $(A \wedge B), (A \vee B), (A \to B), (A \equiv B)$
   has as i.p. A *and* B, **and maybe others (Read On!)**.

3. A is an i.p. of $(\neg A)$, **and maybe has others (Read On!)**. □

2.1.8 Example.

- The i.p. of $((p \vee q) \vee r)$ are $(p \vee q)$ and r
- The i.p. of $(p \vee q)$ are p and q
- The only i.p. of $(\neg\top)$ is $\top$

□

2.1.9 Remark. (Priorities of glue (connectives)) The *priorities* of glue, from left to right in (1) below, go from *strongest* to *weakest*.

$$\neg, \wedge, \vee, \rightarrow, \equiv \tag{1}$$

□

Why do we care? What does "priority" do?

Well, *suppose* we do not want to always write wff down with all the brackets that Definitions 1.5.2 and 2.1.2 require.

Why wouldn't we? For better readability!

Thus we **agree** to judiciously omit brackets in a manner that *we can reinsert them correctly* if we are required to!

That is, we ***agree*** *on how to write formulas* ***sloppily*** *and get away with it!*

Is there any other way to agree *on priorities*?

Yes, BUT: As it is with **any agreement between any two parties**, there can be ONLY ONE agreement.

Remember. We are learning a "programming" language!!

So please *do* follow (1) above and the clarifications that follow below. Anything else will be **wrong**.

The "algorithm" is that whenever two pieces of glue compete for a variable as in, *for example*,

$$\ldots \vee p \wedge \ldots$$

then the *stronger glue wins* (higher priority). In this case it is $\wedge$ that wins and "gets" the p.

This means brackets were intended —and hence are reinserted— this way:

$$\ldots \vee (p \wedge \ldots$$

What if we have the situation

$$\ldots \vee p \vee \ldots \tag{2}$$

i.e., same glue left and right of p?

We have the agreement that all glue is right-associative, that is, in a chain like (2) *the glue on the right wins*! We insert brackets this way:

$$\ldots \vee (p \vee \ldots$$

In particular

$$\neg\neg\neg p$$

means

$$\Big(\neg\big(\neg(\neg p)\big)\Big)$$

$$p \to q \to r \to \bot$$

means

$$(p \to (q \to (r \to \bot)))$$

In $(p \to q) \to r$ cannot remove the brackets; all are needed.

2.1.10 Definition. (Complexity of a wff) The *complexity* of a wff is *the number of occurrences* of connectives (glue) in it. Counting occurrences means that *multiplicity matters* and *counts*! □

2.1.11 Example. Clearly we can compute complexity correctly whether we wrote a formula with all its brackets or not.

For example, the complexity of $p \to \bot \to r$ is 2 whether we wrote it with no brackets or wrote it as Definitions 1.5.2 and 2.1.2 want: $(p \to (\bot \to r))$.

Directly from the definition above, every atomic formula has complexity zero. □

All the theorems (and their corollaries) in this section are **ABOUT** formulas of Boolean logic, and their *FORM*.

They are not *theorems OF* Boolean logic. This concept we have not defined yet!!

Theorems that are ABOUT Boolean logic we call METAtheorems.

2.1.12 Metatheorem. *Every formula A has equal numbers of left and right brackets.*

Proof. Induction on the *complexity*, let's call it n, of A.

1. *Basis.* $n = 0$. Then A has no glue, so it is atomic. But an atomic formula has no left or right brackets!

   Since 0=0 we are good!

2. *Induction Hypothesis*, in short "I.H." Fix an n and assume the statement for all A of complexity $\leq n$.

3. *Induction Step*, in short "I.S.", is for any A of complexity $n+1$. As $n+1 > 0$, A is NOT atomic THEREFORE it has one of *TWO* forms:

(a) A is $(\neg B)$ —where B is a wff.
By I.H. —applicable to B since it has complexity $\leq n$— B has equal number of left and right brackets. Forming A we added one left and one right. So, total left=total right for A.

(b) A is $(B \circ C)$, where we wrote "$\circ$" as a *metasymbol* that stands for any *binary glue* among

$$\wedge, \vee, \rightarrow, \equiv$$

By I.H.

$$Blefts = k, Brights = k, Clefts = k', Crights = k'$$

So, after gluing,

$$BandClefts = k + k', BandCrights = k + k'$$

Overall (after adding external brackets for A),

we have $k + k' + 1$ lefts and $k + k' + 1$ rights. **Bingo**!

□

IMPORTANT! You will note that the induction for the formula A above essentially went like this:

- Prove the property for the atomic formulas $\mathbf{p}, \bot, \top$

  Then we assumed the I.H. that all the i.p. of A have the property.

  and *we proved* (I.S.)

- If A is $(\neg B)$, then A has the property *since the i.p. B does* (**WHY B does?**
  BY I.H. on i.p.)

The technique above is called **Induction on** (the *shape of*) **formulas** and **does not need the concept of complexity.**

This is how we will do it *in our inductions going forward*.

2.1.13 Corollary. *Every* nonempty proper prefix *of a wff A has an excess of left (compared to right) brackets.*

Proof. I will do induction on formulas A.

- *Basis.* A is atomic. Then we are done since A has NO nonempty proper prefix!

  People also say "**then there is nothing to prove**" or "**the statement is vacuously satisfied**".

What just happened here?! Well, I am claiming "the statement is true" and suppose that you are claiming "**the statement is false**".

It is for you to give me a *counterexample* to what I said in order to show that you are right: Namely,

> **You must produce a *nonempty proper prefix* of A that fails the property.**

BUT there is no way! There is NO *nonempty proper prefix* of A!

So I win!

- *Assume* the I.H. that *all the i.p. of A have the property.*

- For the I.S. we examine *ALL possible* *forms* of nonempty proper prefixes. These are:

  1. Case where A is $(\neg B)$. A nonempty proper prefix of A has one of the four **forms** below:

     (a) (Then clearly we have an excess of "(" The I.H. was NOT needed.

     (b) (¬ Then clearly we have an excess of "(" The I.H. again was NOT needed.

     (c) (¬D, where D is an nonempty proper prefix of B. D already has an excess of "(" by the I.H. that applies since B is an i.p. of A.
     So, adding to them the leading red "(" does no harm!

     (d) (¬B Now (2.1.12) B has equal number of lefts and rights. The leading (red)"(" contributes an excess. The I.H. again was NOT needed.

2. A is $(B \circ C)$. A nonempty proper prefix of A has one of the six **forms** below:

   (a) (Then clearly we have an excess of "(" The I.H. was NOT needed.

   (b) $(B'$, where B' is an nonempty proper prefix of B. B' already has an excess of "(" by the I.H. that applies since B is an i.p. of A. So, adding to them the leading "(" does no harm!
   C has balanced brackets so it does not spoil the above.

   (c) $(B$ B has balanced bracket numbers by 2.1.12, thus the leading "(" creates a majority of "(".

   (d) $(B\circ$ As $\circ$ adds no brackets we are done by the previous case.

   (e) $(B\circ C'$ Here B is a formula so it contributes 0 excess. C' is *a nonempty proper prefix of* C and the I.H. applies to the latter as it is an i.p. of A.

   So C' has an excess of "(" and the leading "(" of A helps too.

   (f) $(B \circ C$ Neither B nor C contribute an excess of "(" as both are formulas. The leading red "(" breaks the balance in favour of "(". □

This is easy:

2.1.14 Theorem. *Every formula A begins with an atomic wff, or with a "(".*

Proof. By 2.1.2, A is one of

- Atomic $\mathbf{p}, \bot, \top$
- $(\neg B)$
- $(B \circ C)$ where $\circ \in \{\wedge, \vee, \rightarrow, \equiv\}$

So, in the first case A begins with an atomic wff, and in the other two begins with an "(".

No Induction was used or needed! □

2.1.15 Theorem. (Unique Readability) *The i.p. of any formula A are unique.*

So we can "deconstruct" or "parse" a formula in a **unique way**: A formula is **exactly one of** *atomic*, a *negation* $(\neg B)$, a *disjunction* $(B \vee C)$, a *conjunction* $(B \wedge C)$, an *implication* $(B \to C)$, an *equivalence* $(B \equiv C)$.

Proof.

- Clearly *no atomic formula* can ALSO be "read" *as one of*

  a *negation*, a *disjunction*, a *conjunction*, an *implication*, an *equivalence*

  since the atomic *contains no glue*, but all the others do.

- Can we read a formula A as two *distinct* negations? That is, *using in this proof ONLY "=" as equality of strings*, *can we have*

  $$A = (\neg B) = (\neg C)?$$

  No, since $(\neg B) = (\neg C)$ implies that after we match the *first two symbols* (left to right) then we will continue matching all symbols —by position— until we match all of B with C and finally match the rightmost ")".

- Can we read a formula A as a *negation* **and** as a *disjunction*, or a *conjunction*, or an *implication*, or an *equivalence*? That is, can I have

  $$A = (\neg B) = (C \circ D)?$$

  No, since if we have $(\neg B) = (C \circ D)$, then from left to right the first position is OK (match) but the 2nd is NOT: *C cannot begin with "¬"* (see 2.1.14).

- Can we read a formula A as a $(B \circ C)$ and also *differently* as a $(D \diamond Q)$, *where* $\diamond$ *stands for any binary glue (including* "$\circ$"*)?*

  Let's **assume** *that* we can *and get a* **contradiction**.

  Well, note first that if $(B \circ C) = (D \diamond Q)$ then if we have $B = D$ then this forces $\circ = \diamond$ and hence also that

  $$C) = Q)$$

  which trivially (remove the ending ")") leads to $C = Q$.

  BUT this is not the case that we are worried about.

  So, assume that $B \neq D$. There are **two cases**.

Case 1. B is shorter than D, so is a nonempty proper prefix of D. Then, by 2.1.13, B has an excess of left brackets. But being a wff it also has balanced numbers of left/right brackets. **Contradiction**!

Case 2. D is shorter than B so is a nonempty proper prefix of B. Then, by 2.1.13, D has an excess of left brackets. But being a wff it also has balanced numbers of left/right brackets. **Contradiction**!

□

2.1.16 Remark. Why do we care about unique readability?

OK. Here is an **Example from Arithmetic**. Let us define *arithmetic expressions* (using only $+, \times$) on three numbers, 1, 2, 3. Let us name all such expressions by the *generic name* "E".

Consider the recursive definition below for E, where "::=" is read "is defined as (the rhs)"

1. $E ::= 1$ or $E ::= 2$ or $E ::= 3$

OR

2. $E ::= E + E$.

   The lhs says that to figure the leftmost E out, figure out the *two* rhs E (recursive calls!) and put a glue of "+" between the two.

   I can also say that the definition here says "An (arithmetic) expression E is the sum of TWO arithmetic expressions E".

   Is $E ::= E' + E''$ "more correct?". NO!

   In recursive programming you call the SAME procedure E recursively. Not by some OTHER **names** E' and E''!

OR

3. $E ::= E \times E$.

   The lhs says that to figure the leftmost E out, figure out the *two* rhs E (**recursive calls!**) and put a glue of "$\times$" between the two.

So an example of E is

$$1 + 2 \times 3 \tag{1}$$

What is its value? What are the i.p. of (1)?

Well, *if I say* $\underline{1+2}$ *(+ done first) and* $\underline{3}$, then I get value 9.

If I say $\underline{1}$ *and* $\underline{2 \times 3}$ *(× done first)*, then I get value 7.

Value is slippery! We have ambiguity!

So unique i.p. *IS important*!

□

Sep. 17, 2025

2.2 Boolean Semantics

FORMAL (=syntactically practised) Boolean Logic is about PROVING, SYNTACTICALLY, Boolean THEOREMS. NOT about computing so-called truth values such as "TRUE" or "FALSE".

HOWEVER, it is extremely useful to be familiar and proficient with Boolean SEMANTICS (Concepts that use Boolean *MEANING* such as "TRUE" or "FALSE") since we can use said Semantics when we want to *DISPROVE* a Boolean wff, that is, to *prove* that it has NO PROOF! Is NOT a theorem!

Semantics —or MEANING— hinge on values of wff that we compute! So, TWO questions:

- By what mechanism can wff compute VALUES?
- WHAT ARE these values? 1, 11, 42, $\sqrt{2}$, or something else?

NOTE. The student who has learnt to program a computer also very likely encountered Boolean (wff) values and the techique we employ to compute them for any wff.

These Boolean values are creatures of the METATHEORY —that is, they find themselves OUTSIDE Boolean Logic— since clearly nowhere in the alphabet did we hide away any symbols for values.

Next we must resolve:

> IF we somehow initialised all variables of the alphabet with values, then how do we compute an **overall value of any wff**?

2.3 The Boolean values and initialisation

2.3.1 Definition. A *state* v (or s) —the two usual letters for "state"— is a *function that assigns the value* $\mathbf{f}$ (*false*) or $\mathbf{t}$ (*true*) to *every* Boolean variable —any way we please!— while the *constants* $\bot$ and $\top$, *necessarily, always* get the values $\mathbf{f}$ and $\mathbf{t}$ respectively.

None of these symbols —$v, s, \mathbf{t}, \mathbf{f}$— are in the Boolean logic alphabet $\mathcal{V}$. They are *all metasymbols* in the *meta*theory.

The $\mathbf{f}$ and $\mathbf{t}$ we call *truth values*.

On paper or on the chalk board one usually *underlines* rather than *bolds* —as bolding is cumbersome— so one denotes $\mathbf{f}$ as $\underline{\mathrm{f}}$ and $\mathbf{t}$ as $\underline{\mathrm{t}}$ respectively.

The fact that v gives (assigns) the value $\mathbf{f}$ to the variable q'' is denoted by $v(q'') = \mathbf{f}$. □

Therefore a state v is an *infinite* input/output *table* like the one below

input	output
$\bot$	**f**
$\top$	**t**
p	**t**
q	**f**
$\vdots$	$\vdots$

where *no two rows can have the same input but different outputs*.

In the jargon of MATH1019/1028 the table is what we call a *function*! This observation justifies the notation

$$\underset{\downarrow}{\text{function}} \quad \underset{\downarrow}{\text{output}}$$
$$\underset{v}{\downarrow} \quad (\ \underset{\uparrow}{q''}\ ) = \quad \mathbf{f}$$
$$\text{input}$$

in the last sentence of Definition 2.3.1.

▶ Why an *infinite* table?

Because our *Boolean logic language* has *infinitely many variables* and a state, by definition, assigns a value to *each of them*.

2.3.2 Definition. (Truth tables) In the *metatheory* of Boolean logic —so, outside logic itself— there are five *operations* we are *interested* in that can be applied on the members of the *set of* truth values $\{\mathbf{t}, \mathbf{f}\}$.

Each operation takes its input(s) from the set $\{\mathbf{t}, \mathbf{f}\}$, and its outputs are also in this very same set.

In a very obvious way the 5 operations tell us how values propagate just after we applied glue.

I can say "the operations ape glue in the META domain".

We have one operation for each connective (glue) and in order to keep track of which is formal *and which is* not *we use in the* METATHEORY *(outside Logic) the generic letter F (for "function") subscripted by the name of the corresponding glue.*

These functions of the metatheory are called Boolean functions and are the following.

$$F_{\neg}(x), F_{\vee}(x,y), F_{\wedge}(x,y), F_{\rightarrow}(x,y), F_{\equiv}(x,y)$$

So, "$\vee$" itself does *NOT* operate on *inputs* $\mathbf{f}, \mathbf{t}$.
$F_{\vee}(x,y)$ does. "$\vee$" does not live in the same universe as $\mathbf{f}, \mathbf{t}$.

The behaviour of these F-functions —input/output behaviour, that is— is fully described by the following table that goes by the nickname "*truth table*".

x	y	$F_{\neg}(x)$	$F_{\vee}(x,y)$	$F_{\wedge}(x,y)$	$F_{\rightarrow}(x,y)$	$F_{\equiv}(x,y)$
f	**f**	**t**	**f**	**f**	**t**	**t**
f	**t**	**t**	**t**	**f**	**t**	**f**
t	**f**	**f**	**t**	**f**	**f**	**f**
t	**t**	**f**	**t**	**t**	**t**	**t**

□

One often sees truth values 0 for **f** and 1 for **t** or even the other way around! The metatheory allows many tastes! Not, though, the formal theory!

x	y	$F_{\neg}(x)$	$F_{\vee}(x,y)$	$F_{\wedge}(x,y)$	$F_{\rightarrow}(x,y)$	$F_{\equiv}(x,y)$
0	**0**	**1**	**0**	**0**	**1**	**1**
0	**1**	**1**	**1**	**0**	**1**	**0**
1	**0**	**0**	**1**	**0**	**0**	**0**
1	**1**	**0**	**1**	**1**	**1**	**1**

Chapter 3

What makes our Logic "Classical"

Answer: The behaviour of "$\rightarrow$"!

3.1 States and Truth tables

Refer to the truth table on p.67 and let us discuss the column of $F_{\rightarrow}(x, y)$.

The most "straightforward" entry in this column is arguably, the one for input $(\mathbf{t}, \mathbf{f})$.

This function is describing the truth-value of *implications*, and the x input is the *hypothesis* while the y input is the *conclusion*.

Thus having $F_{\rightarrow}(\mathbf{t}, \mathbf{f}) = \mathbf{f}$ can be **interpreted** as saying that *the implication* ***t → f*** *is FAULTY, i.e.,* **f**, *since we start with a* ***true*** hypothesis *and end up with a* ***false*** conclusion. IMPLICATION MUST PRESERVE TRUTH is our PRINCIPLE.

This *principle indeed supports* the behaviour of $F_{\rightarrow}$ in the other three rows.

For example you would be wrong to tell me: "Hey, $F_{\rightarrow}(\mathbf{f}, \mathbf{t})$ is not right". I will respond: "Hmm! Show me that it does *not* *preserve* truth from hypothesis to conclusion! **You cannot show me a truth here that fails to be preserved!**"

Why "Classical"? This, adopted by Logicians, behaviour of $\rightarrow$ goes all the way back to Aristotle. That's why.

The Classical Implication does not require causality.

So far, *states* give meaning (values) *to atomic formulas only*. Let us *extend* this meaning-giving to *ANY wff*.

3.1.1 Definition. (The value of a wff in some state, v) We *extend* *any* state v to be meaningful *not only for atomic arguments* but also for any *wff* arguments

We will call such an *extension of* v by the same letter, but will "cap" it with a "hat", $\overline{v}$, since it is a different function!

This one, $\overline{v}$, *acts on ANY wff, not only on atomic ones.*

What *IS* an "EXTENSION" of v?

It is a *function* $\overline{v}$ that *on the arguments where* v *is defined* so is $\overline{v}$ and gives the same output!

THIS new function, $\overline{v}$, is obtained by **adding** correct input/output pairs "(**wff, t**)" or "(**wff, f**)" to the v-table.

$\overline{v}$ is **defined** *on more inputs:* On ALL wff found in WFF[a].

[a]Recall that WFF in capitals is the SET of ALL wff

We rest on the shoulders of the "Unique-Readability-Metatheorem":

You see the **significance** of the uniqueness of i.p. at play!!!

The following is like a **Recursive Procedure of Programming!**

$$
\begin{aligned}
\overline{v}(\mathbf{p}) &= v(\mathbf{p}) \\
\overline{v}(\top) &= \mathbf{t} \\
\overline{v}(\bot) &= \mathbf{f} \\
\overline{v}\Big((\neg A)\Big) &= F_{\neg}\Big(\overline{v}(A)\Big) \\
\overline{v}\Big((A \wedge B)\Big) &= F_{\wedge}\Big(\overline{v}(A), \overline{v}(B)\Big) \\
\overline{v}\Big((A \vee B)\Big) &= F_{\vee}\Big(\overline{v}(A), \overline{v}(B)\Big) \\
\overline{v}\Big((A \to B)\Big) &= F_{\to}\Big(\overline{v}(A), \overline{v}(B)\Big) \\
\overline{v}\Big((A \equiv B)\Big) &= F_{\equiv}\Big(\overline{v}(A), \overline{v}(B)\Big)
\end{aligned}
$$

□

Truth tables are more convenient to understand than a bunch of recursive equations; but even easier to misunderstand!

For example the $\vee$-column of the Table in 2.3.2 is often depicted as:

A	B	$A \vee B$
f	**f**	**f**
f	**t**	**t**
t	**f**	**t**
t	**t**	**t**

In recursive definition jargon the above says

$$\overline{v}((A \vee B)) = F_{\vee}(\overline{v}(A), \overline{v}(B))$$

At a glance the table says that to compute the value of $A \vee B$ you just utilise the values of the i.p. A and B as indicated.

The misunderstanding you MUST avoid is this: The two left columns are NOT *values you assign to A and B.*

You can assign values ONLY to ATOMIC formulas!

What these two columns DO say is that *the formulas A and B have each two possible values.*
For each pair of possible values we displayed the outcome value.

3.2 Finite States

We say a variable **p** *occurs in a formula meaning the obvious: It is,* as a string, *a substring —a part— of the formula.*

3.2.1 Metatheorem. *Given a formula A. Suppose that two states, v and s,* **agree** *on* **all** *the variables occurring in A. Then $\overline{v}(A) = \overline{s}(A)$.*

Proof. We do induction on the (**formation of the**) formula A:

1. Case where A is atomic: If it is $\top$ or $\bot$ then $\overline{v}(A) = \overline{s}(A)$ is true.

   If A is $\mathbf{p}$, then

   $$\overline{v}(A) = v(A) \overset{Hypothesis}{=} s(A) = \overline{s}(A)$$

   I.H.: Assume that Claim is true for all i.p. of A.

2. Case where A is $(\neg B)$. We have

   $$\overline{v}(A) = F_{\neg}\Big(\overline{v}(B)\Big) \overset{I.H.}{=} F_{\neg}\Big(\overline{s}(B)\Big) = \overline{s}(A)$$

3. Case where A is $(B \circ C)$.

   $$\overline{v}(A) = F_{\circ}\Big(\overline{v}(B), \overline{v}(C)\Big) \overset{I.H.}{=} F_{\circ}\Big(\overline{s}(B), \overline{s}(C)\Big) = \overline{s}(A)$$

□

3.2.2 Remark. (Finite "appropriate" States) A state v is *by definition an infinite table*.

> By the above METAtheorem, the value of any wff A in a state v *is determined only by the values of* v ON THE VARIABLES OF A, since *any other state* s *that agrees with* v *on said variables gives the same answer.*

Thus, going forward we will be utilising *finite appropriate states* to compute the truth values of any wff.

That is, we discard from the infinite state all the rows that contain variables NOT occurring in the formulas of interest. □

3.2.3 Example. Under no state v can we have

$$((A \to B) \to A) \to A$$

evaluate as **f**. Exercise! □

Sep. 22, 2025

3.3 Tautologies and Tautological Implication

3.3.1 Definition. (Tautologies and other things...)

The expression "ALL states" from here on means "for ALL APPROPRIATE FINITE states".

So, from here on, manipulating states is a FINITE PROCESS.

1. A *Tautology* is a formula A which is true in *all* states. That is, for *all APPROPRIATE* and *FINITE* v, we have $\overline{v}(A) = \mathbf{t}$.

   We write "$\models_{taut} A$" for "*A is a tautology*".

2. A *contradiction* is a formula A such that, for *all* v, we have $\overline{v}(A) = \mathbf{f}$.

   Clearly, for *all* v, we have $\overline{v}(\neg A) = \mathbf{t}$.

Example: $\bot$ is a contradiction according to this definition.

3. A is *satisfiable* iff for $\mathcal{SOME}$ v, we have $\overline{v}(A) = \mathbf{t}$.

   We say that v satisfies A.

▶ *Boolean logic for the user seeks to discover tautologies by proving them.* □

We saw that WFF denotes the set of all (well-formed) formulas.

Capital Greek letters that are different from any Latin capital letter are used to denote arbitrary sets of formulas. Such letters are $\Gamma, \Delta, \Phi, \Psi, \Omega, \Pi, \Sigma$. As always, in the rare and unlikely circumstance you run out of such letters you may use primes and/or (natural number) subscripts.

3.3.2 Definition. (Tautological implication: binary $\models_{taut}$)

1. Let Γ be a set of wff. *We say that v satisfies Γ iff v satisfies —that is, makes* **t**— *every wff in Γ.*

2. We say that Γ *tautologically implies A* —and we write this as $\Gamma \models_{taut} A$— iff *every state v that satisfies Γ* also *satisfies A.*

"Satisfies" is the same as "makes true".

The configuration

$$\Gamma \models_{\text{taut}} A \tag{1}$$

is called a tautological implication claim.

We call Γ *the set of hypotheses* or *premises* of the tautological implication, while A is the *conclusion*.

Instead of $\{A, B, C\} \models_{taut} D$ we write $A, B, C \models_{taut} D$. □

IMPORTANT! The task to verify (1) needs work on our part **ONLY** with v that satisfy Γ.

If there is NO such v then the claim (1) is VACUOUSLY valid!
YOU *cannot* contradict its validity, because to do so you **will *need* a v that *satisfies* Γ but NOT A.**

You have NO COUNTEREXAMPLE.

3.3.3 Example.

(1) If $\models_{\text{taut}} A$, then for any Σ, we have $\Sigma \models_{\text{taut}} A$.

The converse is not valid:

(2) We have $\mathbf{p} \models_{\text{taut}} \mathbf{p} \vee \mathbf{q}$. Indeed, for any v such that $v(\mathbf{p}) = \mathbf{t}$ we compute $\overline{v}(\mathbf{p} \vee \mathbf{q}) = \mathbf{t}$ from the truth table for $\vee$.

Yet, $\mathbf{p} \vee \mathbf{q}$ is NOT a tautology. Just take $v(\mathbf{p}) = v(\mathbf{q}) = \mathbf{f}$

Note also the obvious: $A \models_{taut} A \vee B$, for any wff A and B. Again use the truth table of p.74. □

In view of 3.2.1 we can check all of *satisfiability*, *tautology* status, and *tautological implication* with *finite* Γ *using a finite truth table*.

Examples.

Example 1. $\bot \models_{taut} A$.

Because no v satisfies the lhs of "$\models_{taut}$" so according to Definition, I rest my case.

Example 2. Let us build a truth table for $A \to B \vee A$ and see what we get.

I wrote sloppily, according to our priorities agreement.

I mean $(A \to (B \vee A))$.

We align our part-work under the glue since it is the glue that causes the output.

Here $\to$ is the last (applied) glue. *Under it we write the final results for this formula.*

Since A and B *are not necessarily atomic*, the values under A and B in the table below are *possible* values *NOT assigned values*! *So* $(A \to (B \vee A))$ *is a tautology.*

A	B	A	$\rightarrow$	B	$\vee$	A
f	**f**		**t**		**f**	
f	**t**		**t**		**t**	
t	**f**		**t**		**t**	
t	**t**		**t**		**t**	

Example 3. Here is another tautology. I will verify this by a shortcut method, WITHOUT building a truth table.

I will show

$$\models_{taut} ((A \rightarrow B) \rightarrow A) \rightarrow A \qquad (1)$$

I will do so by arguing that *it is IMPOSSIBLE TO MAKE (1) FALSE*.

- *If (1) is false* then A *is false* and $(A \rightarrow B) \rightarrow A$ *is true*.
- Given the two blue statements above, it *must* be that $A \rightarrow B$ *is false*. *IMPOSSIBLE, since A is false!*

If A is a tautology we say this in symbols as

$$\models_{taut} A \tag{1}$$

Contrast with tautological implication from Γ:

$$\Gamma \models_{taut} A \tag{2}$$

Incidentally, (2) does NOT imply (1): E.g., we have $p \models_{taut} p \vee q$, but $\not\models_{taut} p \vee q$.

On the other hand (1) implies (2) for all Γ choices!

Chapter 4

Substitution and Schemata

4.0.1 Definition. (Substitution in Formulas)

The *META*notation

$$A[\mathbf{p} := B] \tag{1}$$

where A and B are formulas and $\mathbf{p}$ is any variable means

- **As an Action**: "*Find and replace by B ALL occurrences of* $\mathbf{p}$ *in* A".

- **As a Result**: *The STRING resulting from the action* described in the previous bullet. □

1. In the *META*theory of Logic where we use the expression "$[\mathbf{p} := B]$" **we Agree to Give it The Highest priority**: Thus, $A \wedge B[\mathbf{q} := C]$ means $A \wedge \Big(B[\mathbf{q} := C]\Big)$ and $\neg A[\mathbf{p} := B]$ means

$$\neg\Big(A[\mathbf{p} := B]\Big)$$

2. Clearly if $\mathbf{p}$ does NOT occur in A, then the "action" found nothing to replace, so the resulting string —according to (1)— in this case is just A; NO CHANGE.

We observe the following, according to the inductive definition of formulas.

With reference to (1) of page 87, we prove that the result of (1) is a wff.

So how do we do $A[\mathbf{p} := B]$?

Case 1. Say A is atomic. We have three subcases.

- A is $\mathbf{p}$. Then $A[\mathbf{p} := B] = B$
- A is $\mathbf{q}$ —where by $\mathbf{q}$ I denote here a *variable OTHER than the one* $\mathbf{p}$ *stands for*. Then $A[\mathbf{p} := B] = A$ —*no change*.
- A is $\bot$ or $\top$. Then $A[\mathbf{p} := B] = A$ —*no change*.

So in the atomic case, $A[\mathbf{p} := B]$ is ONE OF A or B. IT IS A wff!

According to the recursive definition of wff, we have two more cases for what A is.

Case 2. A is $(\neg C)$. Thus all $\mathbf{p}$'s of A are in C.

Any substitution done happens in C.

The substitution steps are depicted below:

$$A = \overbrace{\boxed{(}}^{3\text{ add ``(''}} \quad \overbrace{\boxed{\neg}}^{2\text{ add }\neg} \quad \overbrace{\underbrace{\boxed{\ldots\mathbf{p}\ldots\mathbf{p}\ldots}}_{C}}^{1\text{ plug }B\text{ in each }\mathbf{p}} \quad \overbrace{\boxed{)}}^{4\text{ add ``)''}}. \qquad (\dagger)$$

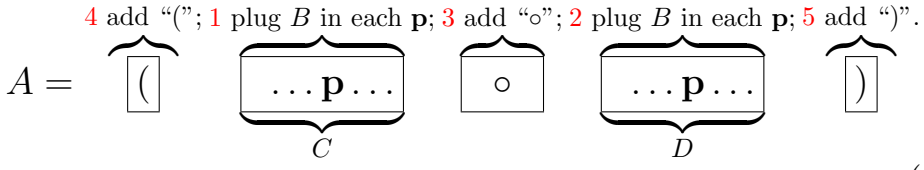
The above actions in $(\dagger)$ have as result

$$A[\mathbf{p} := B] \text{ is } \Big(\neg \underbrace{C[\mathbf{p} := B]}_{by\ I.H.\ a\ wff}\Big)$$

$$\boxed{\text{Hence } A[\mathbf{p} := B] \text{ is a wff in our Case 2.}} \qquad (\dagger')$$

Case 3. A is $(C \circ D)$. Thus all $\mathbf{p}$'s of A are in C and D.

The substitution steps are depicted below:



(‡)

The above actions in (‡) have as result

$$A[\mathbf{p} := B] \text{ is } \Big(\underbrace{C[\mathbf{p} := B]}_{by\ I.H.\ a\ wff} \circ \underbrace{D[\mathbf{p} := B]}_{by\ I.H.\ a\ wff} \Big)$$

Hence $A[\mathbf{p} := B]$ is a wff in our Case 3. (‡′)

We are poised to begin describing the proof system of Boolean logic.

To this end we will need the notation that is called *formula schemata* or *formula "schemas"* (use the latter **only if** you think "schema" is an English word —but it is not!).

4.0.2 Definition. (Schema, Schemata) *Add to the alphabet* $\mathcal{V}$ the following symbols:

1. "[", "]" and ":="

2. All NAMES of formulas: $A, B, C, \ldots$, *with or without primes and/or subscripts*.

3. All *METAsymbols* for variables: $\mathbf{p}, \mathbf{q}, \mathbf{r}$, *with or without primes and/or subscripts*.

Then a *formula schema is a STRING over the* **augmented alphabet**, *which becomes a wff whenever* all *metasymbols of types 2 and 3 above, which occur in the string, are replaced by wff and actual variables (meaning non bold* p, q, r'', q'''_{13}*) respectively, and all actions indicated by substitutions* $[\mathbf{p} := B]$ *are performed*.

A formula that we obtain by the process described in the paragraph above *is called an* **Instance of the Schema**.

□

Three examples of schemata.

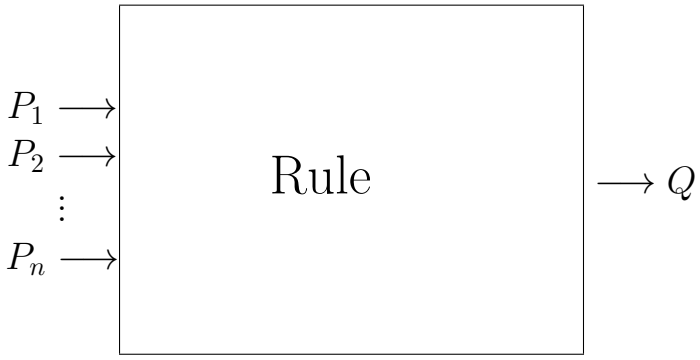
(1) A: This Schema stands (is a placeholder) for a wff! So trivially, *if I plug into A any actual wff, I get that wff as an* **instance***!*

(2) $(A \equiv B)$: Well, whatever formulas I substitute into the (metavariables) A and B I get a wff by the inductive definition of wff.

(3) $A[\mathbf{p} := B]$: We know that if I substitute A and B by actual formulas and $\mathbf{p}$ by an actual Boolean variable I get a wff.

Next stop is Proofs!

In *proofs* we use *Axioms and Rules* (of *Inference*).



A rule as above indicates that in a proof that we have written ALL the wff P_i already, we have the (mathematical) right to write Q *at any proof stage after that!*

It is the habit in the literature to write Rules as *fractions*, not as boxes:

$$\frac{P_1, P_2, \ldots, P_n}{Q} \qquad (R)$$

where *all* of $P_1, \ldots, P_n, Q$ are schemata.

The P_i are *input metavariables* and the Q is the *output metavariable* (only one!)

Thus the "fraction" depicts an action of providing inputs to the P_i and obtaining an output from the metavariable Q.

The P_i are also called *hypotheses* and the Q is called *conclusion*.

More Jargon. We can call the P_i on the numerator of the rule "the *premise(s)*".

The single schema in the denominator we may also call the "*result*".

4.1 Rules and Axioms of Boolean Logic

4.1.1 Definition. (Rules of Inference of Boolean Logic) There are **JUST TWO** up in front ("**called Primary**"):

Rule1

$$\frac{A \equiv B}{\underbrace{C[\mathbf{p} := A]}_{step\ 1} \underbrace{\equiv}_{step\ 3} \underbrace{C[\mathbf{p} := B]}_{step\ 2}} \qquad (Leib[niz])$$

There are NO restrictions in the use of "Leib" —that means Leibniz.

In particular,

(a) **it is NOT required** *that* **p** *actually occurs in* C.

 *If it does* not, *then the denominator is just* $C \equiv C$. 

(b) *The single hypothesis can be ANY equivalence.* —we **do not check it** for "truth".

(c) *For any INPUT we have INFINITELY MANY possible results.*

Rule2 "Equanimity" Rule.

$$\frac{A, A \equiv B}{B} \qquad (Eqn)$$

There are NO restrictions in the use of "Equanimity" other than

> *"A" must be the left part of the equivalence on the numerator.*

Does it matter "left" or "right"? FOR NOW YES!, as we have NO basis to decide otherwise and will NOT be caught "importing" so-called "knowledge" (from other courses) whose validity we did NOT prove in our Logic; YET!!!

□

Sep. 24, 2025

4.1.2 Definition. (Axioms of Boolean Logic) In the following, (1)–(11), A, B, C *name* or *stand for* arbitrary wff.

	Properties of $\equiv$	
Associativity of $\equiv$	$((A \equiv B) \equiv C) \equiv (A \equiv (B \equiv C))$	(1)
Commutativity of $\equiv$	$(A \equiv B) \equiv (B \equiv A)$	(2)
	Properties of $\bot, \top$	
$\top$ and $\bot$	$\top \equiv \bot \equiv \bot$	(3)
	Properties of $\neg$	
Introduction of $\neg$	$\neg A \equiv A \equiv \bot$	(4)
	Properties of $\vee$	
Associativity of $\vee$	$(A \vee B) \vee C \equiv A \vee (B \vee C)$	(5)
Commutativity of $\vee$	$A \vee B \equiv B \vee A$	(6)
Idempotency of $\vee$	$A \vee A \equiv A$	(7)
Distributivity of $\vee$ **over** $\equiv$	$A \vee (B \equiv C) \equiv A \vee B \equiv A \vee C$	(8)
"Excluded Middle"	$A \vee \neg A$	(9)
	Properties of $\wedge$	
"Golden Rule"	$A \wedge B \equiv A \equiv B \equiv A \vee B$	(10)
	Properties of $\rightarrow$	
Implication	$A \rightarrow B \equiv A \vee B \equiv B$	(11)

All of the above (1)–(11) except (3) are schemata for axioms. We call them *Axiom Schemata*, while (3) is an *Axiom*. Each axiom schema above defines *infinitely many axioms* that are its *Instances*.

So our axioms are (3) and all the instances of the Axiom Schemata (1), (2), (4)–(11).

We reserve the Greek letter Λ *for the set of all Axioms of Boolean Logic.*

□

DISCLOSURE. You can verify (Exercise! TRY IT!!!) that each axiom schema (and axiom (3)) is a tautology. The aim is that starting with these tautologies in a proof, and having the Rules PRESERVE TRUTH as we proceed with the proof (WAIT and SEE!), it follows that ANY proof is composed ONLY of true statements.

4.1.3 Definition. (Proofs) Let Γ (could also call it Σ, Θ, Ψ, etc., instead of Γ —there is nothing special about the letter Γ! Use Σ or Δ or Φ if you prefer!!!) be some set of wff.

A *proof* **from HYPOTHESES** Γ is any *finite ordered* sequence of formulas that satisfies the following two *specifications*:

At every step of the *Construction (that we call "Proof")* we *MAY WRITE nothing else except*

Proof 1. *Any ONE wff* from Λ or Γ.

Proof 2. *Any ONE wff A which is the RESULT of an Application* of the rule *Leib* or rule *Eqn* to wff(s)

that appeared in *THIS* proof *before THIS A*.

A proof from hypotheses Γ *is also called a "*Γ*-proof".* □

4.1.4 Remark.

(1) So, *a proof is a totally syntactic construct*, *totally devoid of semantic concepts*.

(2) Γ is a *convenient* set of "*additional hypotheses*".

Syntactically the elements of Γ "behave" like the Axioms from Λ —as it is clear from 4.1.3, item 1— but *semantically they are NOT the same*:

While every member of Λ is a *tautology* by choice,

this *need NOT be the case for the members of* Γ.

(3) Since *every proof* (from some Γ) has *finite length*,

only a finite part of Γ and Λ can EVER appear in some proof.

□ 

4.1.5 Definition. (Theorems) *Any* wff A that *appears in a* Γ*-proof* is called a Γ-theorem.

In particular, any member from Γ or Λ that appears in a proof IS a Γ-theorem.
Remember this!

We also say, "A is a theorem *from* Γ".

In symbols, the sentence "A is a Γ-theorem", is denoted by "$\Gamma \vdash A$".

If $\Gamma = \emptyset$ *then we write* $\vdash A$.

That is, Λ *never appears to the left of the turnstile* "$\vdash$".

We call an A *such that* $\vdash A$ *an absolute or logical theorem.* □

4.1.6 Remark. That A is a Γ-theorem is certified by a Γ-proof like this

$$B_1, \ldots, B_n, A, C_1, \ldots, C_m \tag{1}$$

the sequence (1) obeying the *specifications* of 4.1.3.

Clearly, the sequence (2) below also satisfies the specifications, since each specification for a B_i or A that utilises *rules* refers to formulas *to the left only*.

Thus the sequence (2) is also a Γ-proof of A!

$$B_1, \ldots, B_n, A \tag{2}$$

The bottom line of this story is expressed as either

1. *If you are proving a theorem* A, *just stop as soon as you wrote it down with justification* in a proof!

   OR

2. *A* Γ*-theorem is a wff that appears* **at the END of some proof**.

□ 

Concatenating two Γ*-proofs*

$$A_1, \ldots, A_n$$

and

$$B_1, B_2, \ldots, B_r$$

results in a Γ*-proof.*

Indeed, checking

$$B_1, B_2, \ldots, B_r, A_1, \ldots, A_n$$

from left to right we give EXACTLY the same reasons that we gave for writing the formulas down in each standalone proof*.*

4.1.7 Exercise. How do we do this Exercise?

By providing a Γ-proof IN WHICH our target theorem appears.

(1) $A, B, C \vdash A$, for any wff A
(2) More generally, if $A \in \Sigma$, then $\Sigma \vdash A$
(3) $\vdash B$, for all $B \in \Lambda$

□

4.1.8 Remark. (Hilbert-style Proofs)

A Γ-proof is also called a "*Hilbert-style proof*" —in honour of the great mathematician *David Hilbert*, who was *the first big supporter* of the idea *to use SYNTACTIC (FORMAL) logic as a* TOOL *in order to do CORRECT mathematics*.

We arrange Hilbert proofs *vertically*, one formula per line, *numbered by its position number*, adding "*annotation* " *to the right of the formula we just wrote*, articulating briefly *HOW exactly* we followed the spec of Definition 4.1.3.

Practical Note. If one forgets *numbering* or *annotation*, or that *each line contains ONE wff ONLY*, then this results *in a VERY BAD grade*! :)

□

4.1.9 Example. (Some very simple Hilbert Proofs)

(a) We verify that "$A, A \equiv B \vdash B$" (goes without saying, *for all wff A and B*).

Well, just write a proof of B with "Γ" being $\{A, A \equiv B\}$.

BTW, we indicate a finite "Γ" like $\{A, A \equiv B\}$ without the braces "{ }" when writing it to the left of "$\vdash$".

(1)	A	⟨hypothesis⟩
(2)	$A \equiv B$	⟨hypothesis⟩
(3)	B	⟨(1) + (2) + (Eqn)⟩

Incidentally, *members of Γ are annotated as "**hypotheses**" and going forward we just write "*hyp*".*

Members of Λ we annotate as "Axioms".

Since A and B are arbitrary *undisclosed* wff, *the expression* $A, A \equiv B \vdash B$ *is a Theorem Schema* (*a theorem, no matter what formulas we plug into* A *and* B).

(b) Next verify the Theorem Schema

I will skip in class, but you study this!

$A \equiv B \vdash C[\mathbf{p} := A] \equiv C[\mathbf{p} := B]$

Here you go:

(1)	$A \equiv B$	$\langle$hyp$\rangle$
(2)	$C[\mathbf{p} := A] \equiv C[\mathbf{p} := B]$	$\langle(1)$ + Leib$\rangle$

C can be any wff (and p any actual Boolean variable) so from ONE hypothesis for fixed A and B we can derive an infinite number of theorems of the "shape" $C[\mathbf{p} := A] \equiv C[\mathbf{p} := B]$.

(c) Something more substantial. *Our First Derived Rule!*

We establish the following *Theorem Schema* that we will refer to as *Transitivity* of $\equiv$ —or simply "*Trans*". **How**? *We write a Hilbert-Style proof!*

$$A \equiv B, B \equiv C \vdash A \equiv C \qquad (Trans)$$

(1) $A \equiv B$ $\langle$hyp$\rangle$

(2) $B \equiv C$ $\langle$hyp$\rangle$

(3) $(A \equiv B) \equiv (A \equiv C)$ $\left\langle (2) + (Leib),\ \text{Denom: "}A \equiv \begin{matrix} B \\ \downarrow \\ \mathbf{p} \\ \uparrow \\ C \end{matrix}\text{"};\ \mathbf{p}\ \text{"}fresh\text{"} \right\rangle$

(4) $A \equiv C$ $\langle (1) + (3) + (Eqn) \rangle$

The typesetting *acrobatics* showing the "A" and the "B" that we "feed" in the Leibniz rule variable $\mathbf{p}$ as in annotation of line (3) will not be repeated again.

BTW: *What is* fresh*? Can I* always *have it?* Why *must* $\mathbf{p}$ *be fresh?*

Well, Say A is $\mathbf{p}\wedge\mathbf{q}$*, a simple example with a NON-fresh* $\mathbf{p}$*.*

Then, after feeding B *to* $\mathbf{p}$ *of "*$A \equiv \mathbf{p}$*" the latter wff becomes*

$$B \wedge \mathbf{q} \equiv B$$

which is NOT the SAME STRING AS $A \equiv B$*.*

$$\overbrace{B \wedge q}^{\text{this is NOT } A} \equiv B$$

We should leave A alone; invariant.

(d) *And a Tricky One!* Verify that "$A \equiv A$" is an absolute theorem for all A. That is,

$$\vdash A \equiv A$$

No "HYP" in the proof below!!

(1)	$A \vee A \equiv A$	⟨axiom⟩
(2)	$A \equiv A$	⟨(1) + (*Leib*): $A[\mathbf{p} := A \vee A] \equiv A[\mathbf{p} := A]$ where $\mathbf{p}$ is "fresh"⟩

Can start the proof with *any* known equivalence that is an axiom. For example, line one could contain $\top \equiv \bot \equiv \bot$. □

Sep. 29, 2025

4.1.10 Metatheorem. (Hypothesis Strengthening) If $\Gamma \vdash A$ and $\Gamma \subseteq \Delta$, then also $\Delta \vdash A$.

Proof. *A Γ-proof for A is also a Δ-proof*, since every time we say about a formula B in the proof "legitimate since $B \in \Gamma$" we can say instead "legitimate since $B \in \Delta$". □

4.1.11 Metatheorem. (Transitivity of $\vdash$) Assume $\Gamma \vdash B_1$, $\Gamma \vdash B_2, \ldots, \Gamma \vdash B_n$. Let also $B_1, \ldots, B_n \vdash A$. Then we have $\Gamma \vdash A$.

Also: The Metatheorem Title Could be: "Using *DERIVED RULES*" $B_1, \ldots, B_n \vdash A$.

Derived rules are written as above, as "**Γ-Theorem Assertions**", **NOT as "fractions"**

$$\frac{B_1, \ldots, B_n}{A}$$

Proof.

We have Γ-proofs

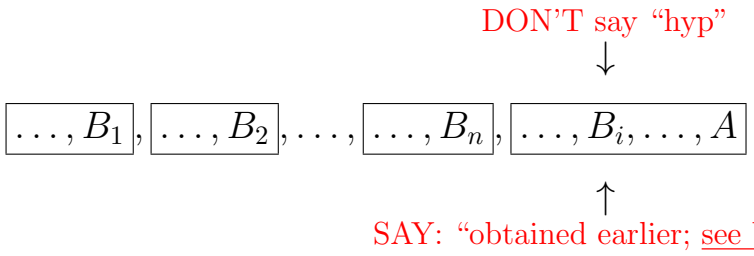
$$\boxed{\ldots, B_1} \quad (1)$$
$$\boxed{\ldots, B_2} \quad (2)$$
$$\vdots$$
$$\boxed{\ldots, B_n} \quad (n)$$

We also have a $\{B_1, \ldots, B_n\}$-proof

$$\boxed{\ldots, B_i, \ldots, A} \qquad (n+1)$$

Concatenate all proofs (1)–(n) (in any order) and to the right of the result glue the proof $(n+1)$.

We have the following proof:



□

So if we view $B_1, \ldots, B_n \vdash A$ as a (derived or "macro" rule) then this "rule" is applicable!

If the B_i are Γ-theorems and $\boxed{B_1, \ldots, B_n \vdash A}$, then we can apply the boxed as a "rule" to obtain the Γ-theorem A.

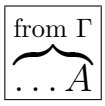
We say "$B_1, \ldots, B_n \vdash A$ is a DERIVED rule".

4.1.12 Metatheorem. *If* $\Gamma \vdash A$ *and also* $\Gamma \cup \{A\} \vdash B$*, then* $\Gamma \vdash B$*.*

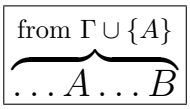
In words, the conclusion says that A *drops out as a hypothesis and we get* $\Gamma \vdash B$*.*

That is, a THEOREM A *can be invoked just like an AXIOM in a proof!*

Proof. We are given *two* proofs:

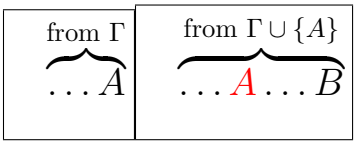


and



When the second box is *standalone*, the justification for A is "hyp".

Now concatenate the two proofs above in the order



Now change all the justifications for the red A in the right box from "hyp" to the exactly same reason you gave to the A in box one —OR, as in the proof of 4.1.11— say about the red A: "obtained earlier in box 1".

Thus, the status of A as "hyp" is removed and B is proved from Γ alone. □

4.1.13 Corollary. If $\Gamma \cup \{A\} \vdash B$ and $\vdash A$, then $\Gamma \vdash B$.

Proof. By hyp strengthening, I have $\Gamma \vdash A$. Now apply the previous metatheorem. □

4.2 Brackets in Chains and Redundant ⊤

4.2.1 Theorem. $A \equiv B \vdash B \equiv A$

Proof.

(1)	$A \equiv B$	⟨hyp⟩
(2)	$(A \equiv B) \equiv (B \equiv A)$	⟨axiom⟩
(3)	$B \equiv A$	⟨(1,2) + Eqn⟩

4.2.2 Theorem. $\vdash (A \equiv (B \equiv C)) \equiv ((A \equiv B) \equiv C)$

NOTE. This is the mirror image of Axiom (1).

Proof.

(1) $((A \equiv B) \equiv C) \equiv (A \equiv (B \equiv C))$ ⟨axiom⟩
(2) $(A \equiv (B \equiv C)) \equiv ((A \equiv B) \equiv C)$ ⟨(1)+4.2.1⟩ □

4.2.3 Remark. Thus, *in a chain of two "≡" we can shift brackets from left to right (axiom) but also right to left (above theorem).*

So it does not matter how brackets are inserted in such chain.

An induction proof on chain length extends this remark to any chain of "≡", of any length.

See course URL, bullet #4 under Notes:
`http://www.cs.yorku.ca/~gt/courses/MATH1090F23/1090.html`.

□ 

4.2.1 The "other EQN" and Redundant ⊤

4.2.4 Theorem. (The "other" (Eqn)**)** $B, A \equiv B \vdash A$

Proof.

(1)	B	⟨hyp⟩
(2)	$A \equiv B$	⟨hyp⟩
(3)	$B \equiv A$	⟨(2) + 4.2.1⟩
(4)	A	⟨(1, 3) + original (Eqn)⟩

□

4.2.5 Corollary. $\vdash \top$

Proof.

(1)	$\top \equiv \bot \equiv \bot$	⟨axiom⟩
(2)	$\bot \equiv \bot$	⟨the "$A \equiv A$" theorem⟩
(3)	$\top$	⟨(1, 2) + (Eqn)⟩

□

4.2.6 Theorem. $\vdash A \equiv A \equiv B \equiv B$

(1) $(A \equiv B \equiv B) \equiv A$ ⟨axiom; brackets **as I please!**⟩
(2) $A \equiv (A \equiv B \equiv B)$ ⟨(1) + 4.2.1⟩ □

NOTE! A possible *VALID* way to perceive (2) above is also as the theorem $(A \equiv A) \equiv (B \equiv B)$.

4.2.7 Corollary. $\vdash \bot \equiv \bot \equiv B \equiv B$ *and* $\vdash A \equiv A \equiv \bot \equiv \bot$

NOTE *absence of brackets in theorem AND corollary!*

4.2.8 Corollary. (Redundant ⊤ Theorem)
$\vdash \top \equiv A \equiv A$ *and* $\vdash A \equiv A \equiv \top$.

Proof.

(1)	$\top \equiv \bot \equiv \bot$	⟨axiom⟩
(2)	$\bot \equiv \bot \equiv A \equiv A$	⟨absolute theorem 4.2.7⟩
(3)	$\top \equiv A \equiv A$	$\langle (Trans) + (1, 2) \rangle$

4.2.9 Metatheorem. (Redundant $\top$ METAtheorem) *For any Γ and A, we have $\Gamma \vdash A$ iff $\Gamma \vdash A \equiv \top$.*

Proof. Say $\Gamma \vdash A$.

Thus

Γ
$\vdots$
(1) A $\langle\Gamma$-theorem$\rangle$
(2) $A \equiv A \equiv \top$ $\langle$Red. $\top$ theorem; 4.2.8$\rangle$
(3) $A \equiv \top$ $\langle$(1, 2) +Eqn$\rangle$

The other direction is similar.

Γ
$\vdots$
(1) $A \equiv \top$ $\langle\Gamma$-theorem$\rangle$
(2) $A \equiv A \equiv \top$ $\langle$Red. $\top$ theorem; 4.2.8$\rangle$
(3) A $\langle$(1, 2) +Eqn$\rangle$

□

4.3 Equational Proofs

Example from high school trigonometry.

Prove that $1 + (\tan x)^2 = (\sec x)^2$ given the identities

$$\tan x = \frac{\sin x}{\cos x} \qquad (i)$$

$$\sec x = \frac{1}{\cos x} \qquad (ii)$$

$$(\sin x)^2 + (\cos x)^2 = 1 \text{ (Pythagoras' Theorem)} \qquad (iii)$$

Equational proof with annotation

$$
\begin{array}{ll}
 & 1 + (\tan x)^2 \\
= & \langle \text{by } (i) \rangle \\
 & 1 + (\sin x / \cos x)^2 \\
= & \langle \text{arithmetic} \rangle \\
 & \dfrac{(\sin x)^2 + (\cos x)^2}{(\cos x)^2} \\
= & \langle \text{by } (iii) \rangle \\
 & \dfrac{1}{(\cos x)^2} \\
= & \langle \text{by } (ii) \rangle \\
 & (\sec x)^2
\end{array}
\qquad (E)
$$

An equational proof in Logic looks IN PRINCIPLE, like:

$$\overbrace{A_1 \equiv A_2}^{\text{reason}}, \overbrace{A_2 \equiv A_3}^{\text{reason}}, \overbrace{A_3 \equiv A_4}^{\text{reason}} \ldots, \overbrace{A_n \equiv A_{n+1}}^{\text{reason}} \tag{1}$$

4.3.1 Metatheorem. (Important Derived Rule!)

$$A_1 \equiv A_2, A_2 \equiv A_3, \ldots, A_n \equiv A_{n+1} \vdash A_1 \equiv A_{n+1} \tag{2}$$

Proof. *By induction on (***left index***)* $n \geq 1$ *using the (derived) rule* $(Trans)$.

1. *Basis* for $n = 1$. We want $A_1 \equiv A_2 \vdash A_1 \equiv A_2$. This is "$X \vdash X$" done! (see 4.1.7).

2. I.H. Assume (2) for fixed unspecified n.

3. I.S. Do the case $n + 1$ for the n we fixed above, so now we want (3) below:

$$A_1 \equiv A_2, A_2 \equiv A_3, \ldots, A_{n+1} \equiv A_{n+2} \vdash A_1 \equiv A_{n+2} \tag{3}$$

Here it goes

Generalised $\equiv$-Transitivity (Derived Rule)

(1)	$A_1 \equiv A_2$	$\langle \text{hyp} \rangle$
(2)	$A_2 \equiv A_3$	$\langle \text{hyp} \rangle$
	$\vdots$	$\vdots$
(n)	$A_n \equiv A_{n+1}$	$\langle \text{hyp} \rangle$
$(n+1)$	$A_1 \equiv A_{n+1}$	$\langle (1+2+\ldots+n) + \text{ I.H.} \rangle$
$(n+2)$	$A_{n+1} \equiv A_{n+2}$	$\langle \text{hyp} \rangle$
$(n+3)$	$A_1 \equiv A_{n+2}$	$\langle n+1, n+2 + \text{Trans} \rangle$

□

All Equational Proofs are based on Metatheorem 4.3.1:

4.3.2 Corollary. *In an Equational proof (from Γ) like the one in (1) of p.129 we have $\Gamma \vdash A_1 \equiv A_{n+1}$.*

Proof. So we have n Γ-proofs, for $i = 1, \ldots, n$,

$$\boxed{\ldots, A_i \equiv A_{i+1}}$$

Concatenate them all to get ONE Γ-proof

$$\overbrace{\boxed{\ldots, A_1 \equiv A_2}}^{\Gamma\text{-proof}} \ldots \overbrace{\boxed{\ldots, A_i \equiv A_{i+1}}}^{\Gamma\text{-proof}} \ldots \overbrace{\boxed{\ldots, A_n \equiv A_{n+1}}}^{\Gamma\text{-proof}}$$

By the DERIVED RULE 4.3.1 the following is a Γ-proof of $A_1 \equiv A_{n+1}$

$$\boxed{\ldots, A_1 \equiv A_2} \ldots \boxed{\ldots, A_i \equiv A_{i+1}} \ldots \boxed{\ldots, A_n \equiv A_{n+1}}, A_1 \equiv A_{n+1}$$

□

4.3.3 Corollary. *In an Equational proof (from Γ) like the one in (1) of p.129 we have $\Gamma \vdash A_1$ iff $\Gamma \vdash A_{n+1}$.*

Proof. *From the above Corollary we have*

$$\Gamma \vdash A_1 \equiv A_{n+1} \qquad (\dagger)$$

Now split the "iff" in two directions:

- IF ($\leftarrow$): So we have

  $$\Gamma \vdash A_{n+1}$$

  This plus ($\dagger$) plus Eqn yield $\Gamma \vdash A_1$.

  So if A_{n+1} is provable, so is A_1.

- ONLY IF ($\rightarrow$): So we have

  $$\Gamma \vdash A_1$$

  This plus ($\dagger$) plus Eqn yield $\Gamma \vdash A_{n+1}$.

□

Oct. 1, 2025

Equational Proof Layout

Successive equivalences like "$A_i \equiv A_{i+1}$ and $A_{i+1} \equiv A_{i+2}$" we write vertically, without repeating the shared formula A_{i+1}.

WITH annotation in $\langle \ldots \rangle$ brackets

$$
\begin{aligned}
& A_1 \\
\equiv\ & \langle \text{annotation} \rangle \\
& A_2 \\
\equiv\ & \langle \text{annotation} \rangle \\
& \vdots \qquad\qquad (ii) \\
& A_{n-1} \\
\equiv\ & \langle \text{annotation} \rangle \\
& A_n \\
\equiv\ & \langle \text{annotation} \rangle \\
& A_{n+1}
\end{aligned}
$$

EXCEPT FOR ONE THING!

(ii) is just ONE FORMULA, namely

$$A_1 \equiv A_2 \equiv \ldots \equiv A_n \equiv A_{n+1}$$

where I can put brackets anywhere I please.

It does NOT say *the same thing as (1) of p.129.*

For example, "$\top \equiv \bot \equiv \bot$" is NOT the same as "$\underbrace{\top \equiv \bot}_{\mathbf{f}} AND \underbrace{\bot \equiv \bot}_{\mathbf{t}}$"

The former (blue) is true but the latter (red) is false.

What do we do?

We introduce a metasymbol for an equivalence that acts ONLY on TWO formulas!

AND

Such equivalences CANNOT be chained to form a SINGLE formula.

The symbol we will use for such UNCHAINABLE equivalences *is "$\Leftrightarrow$" and thus*

"$A \Leftrightarrow B \Leftrightarrow C$" MEANS "$A \equiv B$ AND $B \equiv C$", NOT "$A \equiv B \equiv C$".

We say that "$\Leftrightarrow$" is CONJUNCTIONAL while "$\equiv$" is associative.

So the **final** *layout is:*

$$
\begin{aligned}
& A_1 \\
\Leftrightarrow\ & \langle \text{annotation} \rangle \\
& A_2 \\
\Leftrightarrow\ & \langle \text{annotation} \rangle \\
& \vdots \\
& A_{n-1} \\
\Leftrightarrow\ & \langle \text{annotation} \rangle \\
& A_n \\
\Leftrightarrow\ & \langle \text{annotation} \rangle \\
& A_{n+1}
\end{aligned}
$$

A Lot of Practice Examples Now!

If I a say "I will skip this, but you should do it (do study it)", then indeed you do it and you must know/remember both the proof technique AND the theorem!

You can use the latter in assignments, midterm, exam. As is!

4.3.4 Theorem. $\vdash \neg(A \equiv B) \equiv A \equiv \neg B$

Proof. (Equational)

$$
\begin{array}{ll}
 & \neg(A \equiv B) \\
\Leftrightarrow & \langle \text{axiom} \rangle \\
 & A \equiv B \equiv \bot \\
\Leftrightarrow & \langle (\mathit{Leib}) + \text{axiom: } \neg B \equiv B \equiv \bot; \text{ Denom: } A \equiv \mathbf{p}; \mathbf{p} \text{ fresh} \rangle \\
 & A \equiv \neg B
\end{array}
$$

□

Why do I need Leib above? Why not just use the Axiom? Because I am replacing PART of a wff.

▶ Study this next one, but I will SKIP!

4.3.5 Corollary. $\vdash \neg(A \equiv B) \equiv \neg A \equiv B$

Proof. (Equational)

$\neg(A \equiv B)$
$\Leftrightarrow$ $\langle$axiom$\rangle$
$A \equiv B \equiv \bot$
$\Leftrightarrow$ $\langle(Leib) +$ axiom: $B \equiv \bot \equiv \bot \equiv B$; Denom: $A \equiv \mathbf{p}$; $\mathbf{p}$ fresh$\rangle$
$A \equiv \bot \equiv B$
$\Leftrightarrow$ $\langle(Leib) +$ axiom: $A \equiv \bot \equiv \neg A$; Denom: $\mathbf{q} \equiv B$; $\mathbf{q}$ fresh$\rangle$
$\neg A \equiv B$ □

4.3.6 Theorem. (Double Negation) $\vdash \neg\neg A \equiv A$

Proof. (Equational)

$$
\begin{array}{ll}
& \neg\neg A \\
\Leftrightarrow & \langle \text{axiom } \text{“}\neg X \equiv X \equiv \bot\text{”} \rangle \\
& \neg A \equiv \bot \\
\Leftrightarrow & \langle (Leib) + \text{axiom: } \neg A \equiv A \equiv \bot; \text{ Denom: } \mathbf{p} \equiv \bot \ \rangle \\
& A \equiv \bot \equiv \bot \\
\Leftrightarrow & \langle (Leib) + \text{axiom: } \top \equiv \bot \equiv \bot; \text{ Denom: } A \equiv \mathbf{q}; \ \mathbf{q} \text{ fresh} \rangle \\
& A \equiv \top \\
\Leftrightarrow & \langle \text{red. } \top \text{ thm.} \rangle \\
& A
\end{array}
$$

□

4.3.7 Theorem. $\vdash \top \equiv \neg\bot$

Proof. (Equational)

$$
\begin{aligned}
& \top \\
\Leftrightarrow\ & \langle \text{axiom} \rangle \\
& \bot \equiv \bot \\
\Leftrightarrow\ & \langle \text{axiom} \rangle \\
& \neg\bot
\end{aligned}
$$

□

4.3.8 Theorem. $\vdash \bot \equiv \neg\top$

Proof. (Equational)

$$\begin{aligned} & \neg\top \\ \Leftrightarrow & \langle \text{axiom} \rangle \\ & \top \equiv \bot \\ \Leftrightarrow & \langle \text{red. } \top \rangle \\ & \bot \end{aligned}$$

□

Practical Advise. In Equational Proofs move from the most complex side towards the least complex one.

STARTING WITH WHAT YOU ARE PROVING

4.3.9 Theorem. $\vdash A \vee \top$

Proof.

$A \vee \top$

$\Leftrightarrow$ $\langle (Leib) + \text{axiom } \top \equiv \bot \equiv \bot$; "Denom:" $A \vee \mathbf{p}$; **Mind brackets!**$\rangle$

$A \vee (\bot \equiv \bot)$

$\Leftrightarrow$ $\langle$axiom$\rangle$

$A \vee \bot \equiv A \vee \bot$ Bingo! Recognised an axiom or known theorem. $\square$

Recall about $\equiv$ that, by axiom (1) and a theorem we proved in the NOTES posted in `http://www.cs.yorku.ca/~gt/courses/MATH1090F23/1090.html` (4th bullet), we have that

in a chain of any number of $\equiv$ we may omit brackets.

The same holds for a chain of $\vee$ (and $\wedge$) using *the same kind of proof, in the same link mentioned above.*

That is,

we do not need to show bracketing in a chain of $\vee$ (or one of $\wedge$).

How about *moving* formulas around in such a chain? (*Permuting* them).

It is OK! I prove this for $\vee$-chains HERE. The proof is identical for $\equiv$-chains and $\wedge$-chains (EXERCISE!!)

Prove first this theorem:

$$\vdash B \vee C \vee D \equiv D \vee C \vee B$$

Indeed here is a proof:

$B \vee C \vee \overset{\downarrow}{D}$

$\Leftrightarrow \langle \vee$ commutativity axiom (imagine brackets around $B \vee C)\rangle$

$\overset{\downarrow}{D} \vee B \vee C \qquad (*)$

$\Leftrightarrow \langle (Leib) + \vee$ commutativity axiom. "Denom:" $D \vee \mathbf{p}\rangle$

$D \vee C \vee B$

More generally we CAN DO an arbitrary swap (*not only* the END-FORMULAS), that is, we have the theorem

$$\vdash \boxed{A} \vee B \vee \boxed{C} \vee D \vee \boxed{E} \equiv \boxed{A} \vee D \vee \boxed{C} \vee B \vee \boxed{E}$$

 The boxed formulas may well be long $\vee$-chains! 

Follows by an application of the previous special case:

$$\begin{aligned} & A \vee \overbrace{B \vee C \vee D}^{replace} \vee E \\ \Leftrightarrow\ & \langle (Leib) + \text{equiv. of braced parts. ``Denom:''}\ A \vee \mathbf{p} \vee E \rangle \\ & A \vee \underbrace{D \vee C \vee B}_{by\ this} \vee E \end{aligned}$$

4.3.10 Theorem. $\vdash A \vee \bot \equiv A$

Proofs. (Equational)

This time we work with the entire formula, not just one of the sides of "$\equiv$".


How do we know? **We don't!** It is just a matter of practice.


$$
\begin{aligned}
& A \vee \bot \equiv A \\
\Leftrightarrow\ & \langle (Leib) + \text{idemp. axiom; "Denom:"}\ \ A \vee \bot \equiv \mathbf{p} \rangle \\
& A \vee \bot \equiv A \vee A \\
\Leftrightarrow\ & \langle \text{axiom } \vee \text{ over } \equiv \rangle \\
& A \vee (\bot \equiv A) \\
\Leftrightarrow\ & \langle (Leib) + \text{axiom: } \bot \equiv A \equiv \neg A; \text{ "Denom:"}\ \ A \vee \mathbf{p} \rangle \\
& A \vee \neg A \qquad \textit{Bingo}! \qquad \square
\end{aligned}
$$

Comment. Leib. with "same mouth" **p** used twice in above proof. What about freshness?

Oct. 6, 2025

4.3.11 Theorem. $\vdash A \to B \equiv \neg A \vee B$

Proof.

$$
\begin{array}{ll}
 & A \to B \\
\Leftrightarrow & \langle \text{axiom} \rangle \\
 & A \vee B \equiv B \qquad \blacktriangleleft \text{HERE} \\
\Leftrightarrow & \langle (Leib) + 4.3.10;\ \text{“Denom:”}\ \ A \vee B \equiv \mathbf{p} \rangle \\
 & A \vee B \equiv \bot \vee B \\
\Leftrightarrow & \langle \text{axiom} \rangle \\
 & (A \equiv \bot) \vee B \\
\Leftrightarrow & \langle (Leib) + \text{axiom};\ \text{“Denom:”}\ \ \mathbf{p} \vee B \rangle \\
 & \neg A \vee B
\end{array}
$$

□

4.3.12 Corollary. (IMPORTANT)

$$\vdash \neg A \vee B \equiv A \vee B \equiv B$$

Proof. Start the above proof from spot marked (above) "HERE". □

READ THIS! I will skip. Uses IMPORTANT Corollary above twice.

4.3.13 Theorem. (de Morgan 1)
$\vdash A \wedge B \equiv \neg(\neg A \vee \neg B)$

Proof.

Long but obvious. Start with the most complex side!

$\neg(\neg A \vee \neg B)$
$\Leftrightarrow \langle$axiom$\rangle$
$\neg A \vee \neg B \equiv \bot$
$\Leftrightarrow \langle (Leib) + 4.3.12$; "Denom:" $\mathbf{p} \equiv \bot\rangle$
$A \vee \neg B \equiv \neg B \equiv \bot$
$\Leftrightarrow \langle (Leib) +$ axiom; "Denom:" $A \vee \neg B \equiv \mathbf{p}$ —order does not matter!$\rangle$
$A \vee \neg B \equiv B$
$\Leftrightarrow \langle (Leib) + 4.3.12$; "Denom:" $\mathbf{p} \equiv B\rangle$
$A \vee B \equiv A \equiv B$
$\Leftrightarrow \langle$GR axiom —order does not matter$\rangle$
$A \wedge B$ □

4.3.14 Corollary. (de Morgan 2)

$$\vdash A \vee B \equiv \neg(\neg A \wedge \neg B)$$

Proof. See Text. Better still, EXERCISE!

MORE About "∧"

4.3.15 Theorem. $\vdash A \wedge A \equiv A$

Proof.

$$
\begin{array}{ll}
& A \wedge A \equiv A \\
\Leftrightarrow & \langle \text{GR axiom —order does not matter} \rangle \\
& A \vee A \equiv A \qquad \textit{Bingo!}
\end{array}
\qquad \square
$$

4.3.16 Theorem. $\vdash A \wedge \top \equiv A$

Proof.

$$
\begin{array}{ll}
& A \wedge \top \equiv A \\
\Leftrightarrow & \langle \text{GR axiom} \rangle \\
& A \vee \top \equiv \top \\
\Leftrightarrow & \langle \text{Red. } \top \text{ Thm.} \rangle \\
& A \vee \top \qquad \textit{Bingo!}
\end{array}
\qquad \square
$$

4.3.17 Theorem. $\vdash A \wedge \bot \equiv \bot$

Proof.

$$
\begin{aligned}
& A \wedge \bot \equiv \bot \\
\Leftrightarrow\ & \langle \text{GR axiom} \rangle \\
& A \vee \bot \equiv A \quad \textit{Bingo!}
\end{aligned}
$$

□

▶ READ the following theorem and its proof!
REMEMBER the result!!!

4.3.18 Theorem. (Distributive Laws between $\vee$ and $\wedge$)

(i) $$\vdash A \vee B \wedge C \equiv (A \vee B) \wedge (A \vee C)$$

and

(ii) $$\vdash A \wedge (B \vee C) \equiv A \wedge B \vee A \wedge C$$

The above are written in least parenthesised notation!
It is part of your tool-set!

Proof.

We just prove (i).

$$
\begin{array}{ll}
 & (A \vee B) \wedge (A \vee C) \\
\Leftrightarrow & \langle \text{GR} \rangle \\
 & A \vee B \vee A \vee C \equiv A \vee B \equiv A \vee C \\
\Leftrightarrow & \langle (Leib) + \text{scramble an } \vee\text{-chain; ``Denom:'' } \mathbf{p} \equiv A \vee B \equiv A \vee C \rangle \\
 & A \vee A \vee B \vee C \equiv A \vee B \equiv A \vee C \\
\Leftrightarrow & \langle (Leib) + \text{axiom; ``Denom:'' } \mathbf{p} \vee B \vee C \equiv A \vee B \equiv A \vee C \rangle \\
 & A \vee B \vee C \equiv A \vee B \equiv A \vee C
\end{array}
$$

HERE WE STOP, and try to reach this result from the other side: $A \vee B \wedge C$.

$$
\begin{array}{ll}
 & A \vee B \wedge C \\
\Leftrightarrow & \langle (Leib) + \text{GR; ``Denom:'' } A \vee \mathbf{p} \text{; mind brackets!} \rangle \\
 & A \vee (B \vee C \equiv B \equiv C) \\
\Leftrightarrow & \langle \text{axiom} \rangle \\
 & A \vee B \vee C \equiv A \vee (B \equiv C) \\
\Leftrightarrow & \langle (Leib) + \text{axiom; ``Denom:'' } A \vee B \vee C \equiv \mathbf{p} \rangle \\
 & A \vee B \vee C \equiv A \vee B \equiv A \vee C
\end{array}
$$

□

Until now we only proved absolute theorems Equationally.

▶ How about theorems with HYPOTHESES?

To do so we use the Redundant $\top$ METAtheorem:

$\Gamma \vdash A$ iff $\Gamma \vdash A \equiv \top$

The Technique is demonstrated via Examples!

This "trick" converts a Γ-theorem to an equivalence that is a Γ-theorem.

In particular, If $\Gamma = \{\ldots, A, \ldots\}$, then $\Gamma \vdash A \equiv \top$ **BECAUSE** $\Gamma \vdash A$.

Such equivalences help: They allow the use of Leib!

4.3.19 Example.

(1) $A, B \vdash A \wedge B$

(2) $A \vee A \vdash A$

(3) $A \vdash A \vee B$

(4) $A \wedge B \vdash A$ (Similarly, $A \wedge B \vdash B$)

For (1):

$A \wedge B$
$\Leftrightarrow \langle (Leib) + \text{hyp } B + \text{Red. } \top \text{ META};$ "Denom:" $A \wedge \mathbf{p} \rangle$
$A \wedge \top$
$\Leftrightarrow \langle 4.3.16 \rangle$
A Bingo!

NOTES:
▶ $A, B \vdash B$. Hence $A, B \vdash \boxed{B \equiv \top}$

For (2):

$$\begin{array}{ll} & A \\ \Leftrightarrow & \langle \text{axiom} \rangle \\ & A \vee A \quad \text{Bingo!} \end{array}$$

For (3):

$$\begin{array}{ll} & A \vee B \\ \Leftrightarrow & \langle (Leib) + \text{Hyp } A + \text{Red-}\top\text{-META; ``Denom:''} \ \mathbf{p} \vee B \rangle \\ & \top \vee B \qquad\qquad \langle \text{Bingo!} \rangle \end{array}$$

(4) is a bit trickier:

$$
\begin{array}{ll}
& A \\
\Leftrightarrow & \langle 4.3.16 \rangle \\
& A \wedge \top \\
\Leftrightarrow & \langle (Leib) + \text{Hyp } A \wedge B + \text{Red-}\top\text{-META; ``Denom:'' } A \wedge \mathbf{p} \rangle \\
& A \wedge A \wedge B \\
\Leftrightarrow & \langle (Leib) + 4.3.15; \text{ ``Denom:'' } \mathbf{p} \wedge B \rangle \\
& A \wedge B \qquad \text{Bingo!}
\end{array}
$$

□

I SKIP AHEAD TO THE CUT-RULE! You study 4.3.20 proof below :)

4.3.20 Metatheorem. (Hypothesis splitting/merging)
For any wff A, B, C *and hypotheses* Γ, *we have* $\Gamma \cup \{A, B\} \vdash C$ *iff* $\Gamma \cup \{A \wedge B\} \vdash C$.

Proof. (Hilbert-style)
(I) *ASSUME* $\Gamma \cup \{A, B\} \vdash C$ and *PROVE* $\Gamma \cup \{A \wedge B\} \vdash C$.

So, armed with Γ and $A \wedge B$ as *hypotheses* I have to prove C. OK, start!

(1)	$A \wedge B$	$\langle$hyp$\rangle$
(2)	A	$\langle$(1) + $A \wedge B \vdash A$ rule $\rangle$
(3)	B	$\langle$(1) + $A \wedge B \vdash B$ rule $\rangle$
(4)	C	$\langle$using HYP Γ + (2) and (3) $\rangle$

(II) *ASSUME* $\Gamma \cup \{A \wedge B\} \vdash C$ and *PROVE* $\Gamma \cup \{A, B\} \vdash C$. *So, my HYPs are* Γ *and* A *and* B.

(1)	A	$\langle$hyp$\rangle$
(2)	B	$\langle$hyp$\rangle$
(3)	$A \wedge B$	$\langle$(1, 2) + $A, B \vdash A \wedge B$ rule $\rangle$
(4)	C	$\langle$using HYP Γ + (3) $\rangle$

□ 

4.3.21 Theorem. (Cut Rule) $A \vee B, \neg A \vee C \vdash B \vee C$

Proof. We start with an AUXILIARY theorem —a Lemma— which makes the most complex hypothesis $\neg A \vee C$ usable (turns it into an EQUIVALENCE).

$$\begin{aligned} & \neg A \vee C \\ \Leftrightarrow & \langle \text{how to lose a NOT} \rangle \\ & A \vee C \equiv C \end{aligned}$$

Since $\neg A \vee C$ is a HYP hence also a THEOREM, the same is true for $A \vee C \equiv C$ from the Equational proof above. Remember this below!

$$\begin{aligned} & B \vee C \\ \Leftrightarrow & \langle (Leib) + \text{Lemma; “Denom:” } B \vee \mathbf{p} \rangle \\ & B \vee (A \vee C) \\ \Leftrightarrow & \langle \text{inserting brackets to our advantage AND swapping wff} \rangle \\ & (A \vee B) \vee C \\ \Leftrightarrow & \langle (Leib) + \text{HYP } A \vee B + \text{Red-}\top\text{-Meta; “Denom:” } \mathbf{p} \vee C \rangle \\ & \top \vee C \qquad \text{Bingo!} \end{aligned}$$

□

READ ME! SPECIAL CASES of CUT:

4.3.22 Theorem. (Modus Ponens) $A, A \to B \vdash B$

Proof.

$$
\begin{array}{ll}
 & A \to B \\
\Leftrightarrow & \langle \neg\vee\text{-theorem} \rangle \\
 & \neg A \vee B \\
\Leftrightarrow & \langle (Leib) + \text{hyp } A + \text{Red-}\top\text{-META; ``Denom:'' } \neg\mathbf{p} \vee B \rangle \\
 & \neg\top \vee B \\
\Leftrightarrow & \langle (Leib) + \text{theorem from class; ``Denom:'' } \mathbf{p} \vee B \rangle \\
 & \bot \vee B \\
\Leftrightarrow & \langle \text{thm from class} \rangle \\
 & B
\end{array}
$$

□

4.3.23 Corollary. *Alternatively: The above says*

$$A, \neg A \vee B \vdash B$$

Well, Hilbert:

1)	A	$\langle$hyp$\rangle$
2)	$\neg A \vee B$	$\langle$hyp$\rangle$
3)	$A \vee B$	$\langle 1 +$ rule from class/NOTEs (expansion)$\rangle$
4)	$B \vee B$	$\langle 2 + 3 +$ CUT$\rangle$
5)	$B \vee B \equiv B$	$\langle$axiom$\rangle$
6)	B	$\langle 4 + 5 +$ Eqn$\rangle$

4.3.24 Corollary. $A \vee B, \neg A \vdash B$

Proof. Apply the rule $\neg A \vdash \neg A \vee B$. Now invoke 4.3.23 (here we have $\neg A$ instead of A). □

4.3.25 Corollary. $A, \neg A \vdash \bot$

Proof. Hilbert-style.

(1) A $\langle$hyp$\rangle$
(2) $\neg A$ $\langle$hyp$\rangle$
(3) $A \vee \bot$ $\langle 1 +$ rule $X \vdash X \vee Y\rangle$
(4) $\neg A \vee \bot$ $\langle 2 +$ rule $X \vdash X \vee Y\rangle$
(5) $\bot \vee \bot$ $\langle 3 + 4 +$ CUT$\rangle$

Reduce the two $\bot$ via idempotent axiom.

Can you do the above Equationally without using CUT? □

SKIP this PROOF, but memorise the result!

4.3.26 Corollary. (Transitivity of $\to$) $A \to B, B \to C \vdash A \to C$

Proof. (Hilbert style)

(1)	$A \to B$	$\langle$hyp$\rangle$
(2)	$B \to C$	$\langle$hyp$\rangle$
(3)	$A \to B \equiv \neg A \vee B$	$\langle\neg\vee$ thm$\rangle$
(4)	$B \to C \equiv \neg B \vee C$	$\langle\neg\vee$ thm$\rangle$
(5)	$\neg A \vee B$	$\langle(1, 3) + (Eqn)\rangle$
(6)	$\neg B \vee C$	$\langle(2, 4) + (Eqn)\rangle$
(7)	$\neg A \vee C$	$\langle(5, 6) +$ CUT$\rangle$

□

The last line is provably equivalent to $A \to C$ by the $\neg\vee$ theorem.

Chapter 5

Post's Theorem and the Deduction Theorem

Oct. 8, 2025

This Chapter is about the *Soundness* and *Completeness* (the latter is also known as "Post's Theorem") in Boolean logic.

5.1 Soundness of Boolean Logic

MEMORISE RESULTS 1.–3. (but SKIP PROOFS):

1. SOUNDNESS of Boolean Logic.
2. POST's THEOREM (COMPLETENESS of Boolean Logic).
3. DEDUCTION THEOREM.

Soundness is the Property expressed by the statement of the *metatheory* below —which in English says "**Boolean Logic tells ONLY the truth**":

$$\text{If } \Gamma \vdash A, \text{ then } \Gamma \models_{taut} A \tag{1}$$

5.1.1 Definition. The statement "Boolean logic is Sound" means that Boolean logic satisfies (1).

For $\Gamma = \emptyset$ it means this:

$$\text{If } \vdash A, \text{ then } \models_{taut} A \tag{2}$$

□

SKIP TO STATEMENT OF POST'S THEOREM (5.1.2) AND EXAMPLE BEFORE IT.

5.1.2 Example. Soundness allows us to disprove formulas: To show they are NOT theorems.

- The statement "$\vdash \mathbf{p}$" is false. If this were true, then $\mathbf{p}$ would be a tautology!
- The statement "$\vdash \bot$" is false! Because $\bot$ is not a tautology!
- The statement "$p \vdash p \wedge q$" is false. Because if it were true I'd have to have $p \models_{taut} p \wedge q$.

  Not so: Take a state s such that $s(p) = \mathbf{t}$ and $s(q) = \mathbf{f}$. □

5.2 Completeness of Boolean logic ("Post's Theorem")

(1) Post's theorem: If $\Gamma \models_{\text{taut}} A$, then $\Gamma \vdash A$

The *Deduction Theorem — "1st red statement below implies the second"* is a Corollary of (1) above:

(2) If $\Gamma, A \vdash B$, then $\Gamma, A \models_{taut} B$ (Soundness), HENCE $\Gamma \models_{taut} A \to B$, HENCE (Post), $\Gamma \vdash A \to B$.

(1) SPECIAL CASE (Γ finite):

$$\text{IF } \overbrace{A_1, \ldots, A_n}^{\Gamma} \models_{taut} B \text{ THEN } A_1, \ldots, A_n \vdash B$$

That is,

Every $\models_{taut}$ SCHEMA can be USED as a DERIVED RULE.

(2) above is the way practicing mathematicians prove theorems like $\Gamma \vdash A \to B$ in their everyday work.

5.3 Deduction Theorem and Proof by Contradiction

5.3.1 Metatheorem. (The Deduction Theorem) *If* $\Gamma, A \vdash B$, *then* $\Gamma \vdash A \to B$, *where "*Γ, A*" means "all the assumptions in* Γ, *plus the assumption* A*"* (in set notation this would be $\Gamma \cup \{A\}$).*

*We can also write $\Gamma + A$.

The mathematician, or indeed the mathematics practitioner, uses the Deduction theorem all the time, without stopping to think about it. Metatheorem 5.3.1 above makes an honest person of such a mathematician or practitioner.

The everyday "style" of applying the Metatheorem goes like this:

Say we have all sorts of assumptions and we want, *under these assumptions*, to "prove" that "if A, then B" (verbose form of "$A \to B$").

We start by ***adding*** A *to our assumptions*, often with the words, "*Assume* A". We then proceed and prove *just* B (not $A \to B$), and at that point we rest our case.

Thus, we may view an application of the Deduction theorem as a simplification of the proof-task. It allows us to "split" an implication $A \to B$ that we want to prove, moving its premise to join our other assumptions. We now have to prove a *simpler formula*, B, with the help of *stronger* assumptions (that is, all we knew so far, plus A). That often makes our task so much easier!

An Example. Prove

$$\vdash (A \to B) \to A \vee C \to B \vee C$$

By DThm, suffices to prove

$$A \to B \vdash A \vee C \to B \vee C$$

instead.

Again By DThm, suffices to prove

$$A \to B, A \vee C \vdash B \vee C$$

instead.

Let's do it:

1.	$A \to B$	$\langle$hyp$\rangle$
2.	$A \vee C$	$\langle$hyp$\rangle$
3.	$A \to B \equiv \neg A \vee B$	$\langle\neg\vee$ thm$\rangle$
4.	$\neg A \vee B$	$\langle 1 + 3 +$ Eqn$\rangle$
5.	$B \vee C$	$\langle 2 + 4 +$ Cut$\rangle$

□

5.3.2 Definition. A set of formulas Γ is *inconsistent* or *contradictory* iff Γ *proves EVERY (ALL) A in* WFF. □

Why "contradictory"? For if Γ proves *everything*, then it also proves the *contradiction* $\mathbf{p} \wedge \neg \mathbf{p}$.

5.3.3 Lemma. Γ *is inconsistent iff* $\Gamma \vdash \bot$

Proof. *only if*-part. If Γ is as in 5.3.2, then, in particular, it proves $\bot$ since the latter is a well formed formula.

if-part. Say, conversely, that we have

$$\Gamma \vdash \bot \tag{9}$$

Let now A be any formula in WFF whatsoever. We have

$$\bot \models_{taut} A \tag{10}$$

Pause. Do you believe (10)?

By Post (5.2), $\Gamma \vdash A$ follows from (9) and (10). □

5.3.4 Metatheorem. (Proof by contradiction)
$\Gamma \vdash A$ *IFF* $\Gamma \cup \{\neg A\}$ *is inconsistent.*

Proof. *IF.* So let (by 5.3.3)

$$\Gamma, \neg A \vdash \bot$$

Hence

$$\Gamma \vdash \neg A \to \bot \tag{1}$$

by the Deduction theorem. However $\neg A \to \bot \models_{taut} A$, hence, by (1)-in-the-box, and (1) above, $\Gamma \vdash A$.

ONLY IF. Let $\Gamma \vdash A$. Prove $\Gamma, \neg A \vdash \bot$.

Indeed,

$$\Gamma, \neg A \vdash A \tag{$*$}$$

by hyp. strengthening.

By Def. of Γ-proof we also have

$$\Gamma, \neg A \vdash \neg A \tag{$**$}$$

By $A, \neg A \models_{taut} \bot$ and $(*), (**)$ we get $\Gamma, \neg A \vdash \bot$ □

.

5.3.4 legitimises the tool of "proof by contradiction" that goes all the way back to the ancient Greek mathematicians: To prove A assume instead the "opposite", $\neg A$. Proceed then to obtain a contradiction. This being accomplished, it is as good as having proved A.

Chapter 6

Resolution

Proof by Resolution is an easy and self-documenting 2-dimensional proof style.

It is essentially a Hilbert style proof that needs no numbering due to its graphical presentation, where the *annotation is depicted by drawing certain lines*.

The technique is used in "automatic theorem proving", i.e., special computer systems (programs) that prove theorems.

It is based on the *proof by contradiction metatheorem* 5.3.4, namely:

$$\Gamma, \neg A \vdash \bot \tag{1}$$

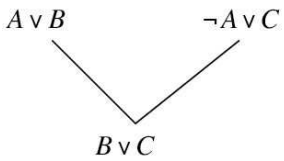
iff

$$\Gamma \vdash A \tag{2}$$

Thus, instead of proving (2) prove (1).

(1) is proved using (almost) exclusively the CUT Rule.

The *self-annotating* diagram below says "apply the CUT rule to premises $A \vee B$ and $\neg A \vee C$ to obtain $B \vee C$".



The technique can be best learnt via examples:

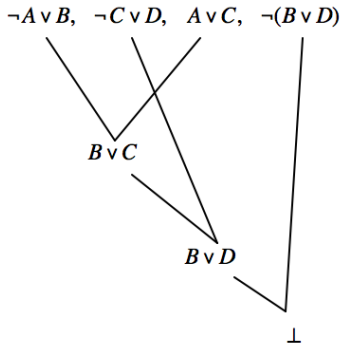
6.0.1 Example. Use Resolution to prove (1) below:

$$A \to B, C \to D \vdash A \vee C \to B \vee D \tag{1}$$

by DThm prove instead:

$$A \to B, C \to D, A \vee C \vdash B \vee D$$

By 4.3.25 prove instead that the "Γ" in the top line below proves $\bot$



□

Oct. 20, 2025

6.0.2 Example. Next prove

$$\vdash (A \to (B \to C)) \to ((A \to B) \to (A \to C))$$

By the DThm prove instead

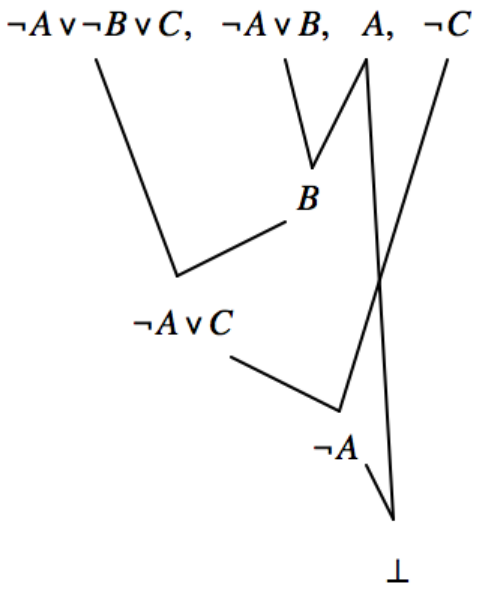
$$A \to (B \to C) \vdash (A \to B) \to (A \to C)$$

Two more applications of the DThm simplify what we will prove into the following:

$$A \to (B \to C), A \to B, A \vdash C$$

By 4.3.25, prove instead that $\Gamma \vdash \bot$ where

$$\Gamma = \{\neg A \vee \neg B \vee C, \neg A \vee B, A, \neg C\}$$



□

6.0.3 Example. *Annotating hypothesis splitting* and equivalence graphically. We *do not annotate* the equivalence or split lines any more than we annotate the CUT lines!

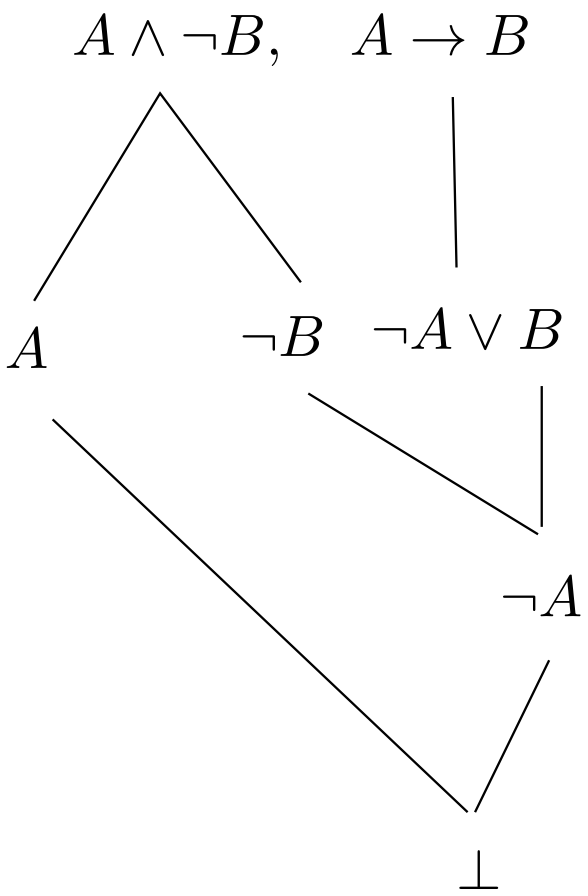
Prove

$$\vdash (A \wedge \neg B) \to \neg(A \to B)$$

By DThm do instead: $A \wedge \neg B \vdash \neg(A \to B)$.

By 4.3.25 do instead

$$A \wedge \neg B, A \to B \vdash \bot$$

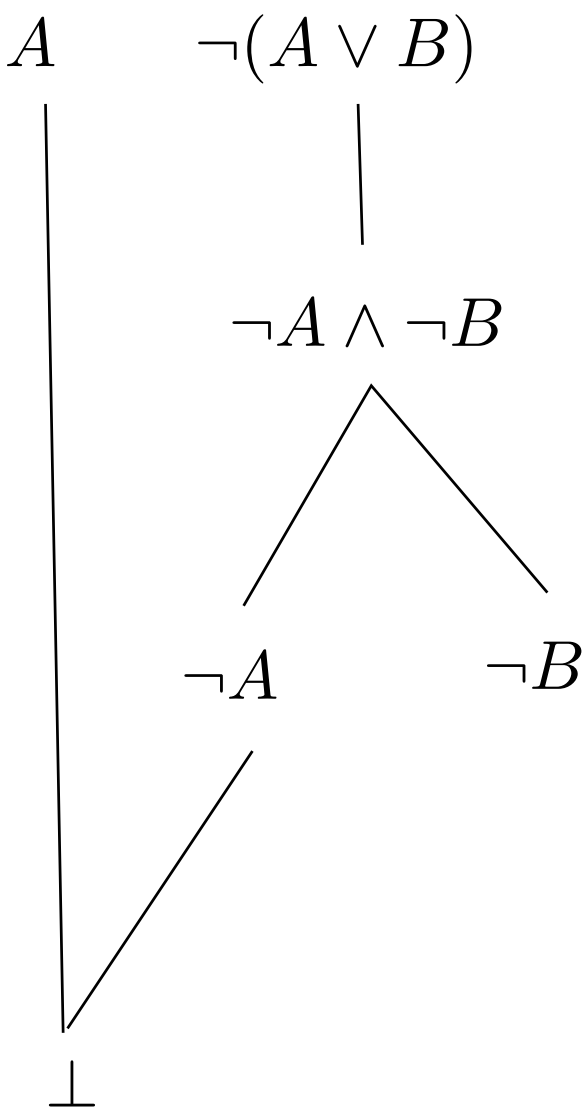


6.0.4 Example. *Annotating hypothesis splitting* and equivalence graphically. We *do not annotate* the equivalence or split lines any more than we annotate the CUT lines!

Prove

$$A \vdash A \vee B$$

By Proof by Contra do instead: $A, \neg(A \vee B) \vdash \bot$.


A
¬(A ∨ B)
¬A ∧ ¬B
¬A
¬B
⊥

□

Chapter 7

Predicate Logic

Oct. 27, 2025

Extending Boolean Logic

Boolean Logic can deal only with the Boolean glue: properties and behaviour.

Can certify tautologies, but it misses *many other truths* as we will see, *like $x = x$ where x stands for a mathematical object like a matrix, string, array, number, etc.*

One of the obvious reasons is that Boolean logic cannot even "see" or "speak" about mathematical objects.

If it cannot see or speak about them, then naturally cannot reason about them either!

E.g, we *cannot even state* inside Boolean logic the sentence "every natural number greater than 1 has a prime factor".

Boolean Logic *does not know* what "every" means or what a "number" is, what "natural" means, what is "1", what "greater" means, what "prime" is, or what "factor" is.

In fact it is worse than not "knowing": It cannot even say any one of the concepts listed above.

Its alphabet and language are extremely limited.

We need a richer language!

7.0.1 Example. Look at these two math statements. The first says that *two sets are equal* IF *they have the same elements*. The second says that *any object is equal to itself*.

We read "$(\forall x)$" below as "*for all values of* x", usually said MORE SIMPLY as, "*for all* x".

$$(\forall y)(\forall z)\Big((\forall x)(x \in y \equiv x \in z) \to y = z\Big) \tag{1}$$

and

$$x = x \tag{2}$$

Boolean Logic is a *very high level* (= very non-detailed) abstraction of Mathematics.

Since Boolean Logic cannot see object variables x, y, z, cannot see $\forall$ or $=$, *nor can penetrate inside the so-called "scope" of* $(\forall z)$ —that is, the big brackets above— it myopically *understands* (*perceives*) each of (1) and (2) as atomic statements p and q (not seeing inside the scope it sees NO "glue").

Thus Boolean logic, if forced to opine about the above it will say none of the above is a theorem (by soundness).

Yet, (1) is an axiom of *Set Theory* and (2) is an *axiom in ALL mathematics*.

Says: "Every object is equal to itself." □

Enter First-Order Logic or Predicate Logic.

Predicate logic is *the language AND logic* of mathematics and mathematical sciences.

In it we CAN "speak" (1) and (2) above and reason about them.

7.1 The language of First-Order Logic

What symbols are *absolutely necessary* to *include* in the Alphabet, $\mathcal{V}_1$ —the subscript "1" for "1st-order"— of Predicate Logic?

Well, let us enumerate:

7.1.1 Definition. (The 1st-order alphabet; first part)

1. First of all, we are *EXTENDING*, *NOT* discarding, *Boolean Logic*. So we include in $\mathcal{V}_1$ *all of Boolean Logic's symbols*

$$\mathbf{p}, \bot, \top, (,), \neg, \wedge, \vee, \rightarrow, \equiv$$

   where $\mathbf{p}$ stands for **any** of the infinitely many Boolean variables.

2. Then we need *object variables* —that is, variables that stand for *mathematical objects*— x, y, z, u, v, w **with or without primes or subscripts**. These are infinitely many.

   Metanotation that stands for any of them will be bold face, but using the same letters with or without primes or subscripts: $\mathbf{x}, \mathbf{x}''_{\mathbf{5}}, \mathbf{y}, \mathbf{w}'''_{\mathbf{123}}$, etc.

3. *Equality* between mathematical objects: $=$

4. *New glue*: $\forall$

   We call this glue *universal quantifier*. It is pronounced "for all".

Is that all? No. *But let's motivate with two examples.* □

7.1.2 Example. (Set theory) The language of set theory needs also a binary relation or *predicate* up in front: Denoted by "$\in$". BUT nothing else.

All else is "*manufactured*" in the theory, that is, introduced by definitions.

The manufactured symbols include *constants* like our familiar $\mathbb{N}$ (the *set of natural numbers*, albeit set theorists often prefer the symbol "ω"), our familiar *constant* "$\emptyset$" (the empty set).

Also include *functions* like $\cup, \cap$ and relations or *predicates* like $\subset, \subseteq$.

So set theory needs no constants or functions up in front to start "operating" (proving theorems, that is). □

7.1.3 Example. (Number theory) The language of Number theory —also called *Peano arithmetic*— needs —in order to get started:

- A *constant*, the *number zero*: 0
- A binary *predicate* ("less than"): $<$
- A unary *function*: "S". (This, informally/intuitively is the "successor function" which with input x produces output $x+1$.)
- Two binary *functions*, "$+, \times$" with the obvious meaning.

All else is "*manufactured*" in the theory, that is, *introduced by definitions*.

The manufactured symbols include *constants* like our familiar $1, 2, 1000234000785$:

$$S0, SS0, \quad \overbrace{SS\ldots S}^{1000234000785\ S\ \text{symbols}} \quad 0$$

We can also manufacture *functions* like x^y, $\lfloor x/y \rfloor$ and more relations or *predicates* like $\leq$. □

We will do logic for the user, that is, we are aiming to *teach the USE of logic*.

But will do so *without having to learn and do set theory or number theory or any specific mathematical theory* (geometry, algebra, etc.).

So equipped with our observations from the examples above, we note that various theories start up with *DIFFERENT* sets of constants, functions and predicates —according to their specific needs.

So we will complete the Definition 7.1.1 in a *UNIFIED* way that *APPLIES TO ANY AREA OF MATHEMATICAL APPLICATION*.

7.1.4 Definition. (The 1st-order alphabet; part 2) Our 1st-order alphabet also includes the following symbols

(1) Symbols for zero or more *constants*. *Generically*, we use a, b, c, d with or without primes or subscripts for constants.

(2) Symbols for zero or more *functions*. *Generically* we use f, g, h with or without primes or subscripts for functions.

Each such symbol will have the need for a certain number of arguments, this number called the function's "*arity*" (must be ≥ 1). For example, S has arity 1; it is unary. Each of $+, \times$ have arity two; they are binary.

You see where the word "arity" comes from?

(3) Symbols for zero or more *predicates*, *generically* denoted as ϕ ("$f\overline{e}$", as in "see"), ψ ("$ps\overline{e}$"), with or without primes or subscripts.

Each predicate symbol will have the need for a certain number of arguments called its "*arity*" (must be ≥ 1). For example, $<$ has arity 2. □

The first-order *LANGUAGE* is a set of strings of two types —*terms* and *formulas*— over the *alphabet* 7.1.1 AND 7.1.4.

By now we should feel comfortable with *first-order inductive definitions.*

In fact we gave inductive definitions of *first-order Boolean formulas* and used it quite a bit, but also more recently gave an inductive definition of Boolean *proofs*.

Thus we inductively introduce first-order Terms that denote objects —the notation is that of function calls— and first-order formulas, that denote statements, in two separate definitions.

First terms:

7.1.5 Definition. (Terms)
A term is a string over the alphabet $\mathcal{V}_1$ that satisfies ONE of:

(1) It is just an *object variable* **x** (recall that **x** is metanotation and stands for *any* object variable).

BTW, we drop the qualifier "object" from "object variable" from now on, but *RETAIN* the qualifier "Boolean" in "Boolean variable".

(2) An *object constant* a (this stands for any constant —generically).

BTW, we ALSO drop the qualifier "object" from "object constant" from now on, but *RETAIN* the qualifier "Boolean" in "Boolean constant".

(3) **General case**. It is a string of the form $ft_1t_2\ldots t_n$ where the function symbol f has *arity* n and the t_i **are** (I mean, STAND FOR) terms.

> As noted already, objects —with the exception of trivial ones, like variables and constants, these are denoted by function calls.
> Surprised? Function calls do return as values objects!

We will denote arbitrary terms generically by the metasymbols t, s *with or without primes or subscripts* just as we did above. □

We will often abuse notation and write "$f(t_1, t_2, \ldots, t_n)$" for "$ft_1t_2 \ldots t_n$".

This is one (rare) case where *the human eye prefers extra brackets*! Be sure to note that the comma "," is not in our alphabet!

Examples from number theory.
$x, 0$ are terms. $x+0$ is a term (abuse of the actual "$+x0$" notation).

$(x + y) \times z$ is a term (abuse of the actual $\times + xyz$).

With the concept of terms out of the way we now define 1st-order formulas:

First the Atomic Case:

Reminder: Arbitrary (non specific) terms are denoted by the letters t and s with or without subscripts or primes.

7.1.6 Definition. (1st-order Atomic formulas) The following are the *atomic —that is, glue-less— formulas of 1st-order logic*:

(i) Any *Boolean* atomic formula.

(ii) The *expression* (*string*) "$t = s$", for any choice of t and s (probably, the t and s name the same term).

(iii) For any predicate ϕ of arity n, and any n terms $t_1, t_2, \ldots, t_n$, the string "$\phi t_1 t_2 \ldots t_n$".

We denote *the set of all atomic formulas* here defined by **AF**. □

In practice, we prefer writing $x < y$ (infix) rather than $< xy$ (prefix)

7.1.7 Remark.

(1) As in the case of "complex" terms $ft_1t_2\dots t_n$, we often abuse notation using "$\phi(t_1, t_2, \dots, t_n)$" in place of the correctly written "$\phi t_1 t_2 \dots t_n$".

(2) The symbol "=" is a binary predicate and is always written as it is here (never "ϕ, ψ").

(3) We absolutely NEVER confuse "=" with the Boolean "glue" "$\equiv$".

They are more different than apples and oranges!

□ 

7.1.8 Definition. (1st-order formulas) A first-order formula A — or **wff** A— is one of

We let context fend for us as to *what formulas we have in mind when we say "wff": 1st Order or Boolean?*

From here on it is 1st-order ones!

If we want to talk about Boolean wff we *WILL ALWAYS USE* the qualifier "Boolean"!

(1) A *member of 1st-order* **AF** *set* —in particular it could be a *Boolean* atomic wff!

(2) $(\neg B)$ if B is a wff.

(3) $(B \circ C)$ if B *and* C are wff, and $\circ$ is one of $\wedge, \vee, \rightarrow, \equiv$.

(4) $\Big((\forall \mathbf{x})B\Big)$, where B is a wff and $\mathbf{x}$ any variable.

TWO things: (1) we already agreed that "variable" means *object variable* otherwise I'd say "Boolean variable". (2) *Nowhere in the definition —of item (4)— is required that* $\mathbf{x}$ *occurs in* B *as a sub-string*.

We call "$\forall$" the *universal quantifier*.

The configuration $(\forall \mathbf{x})$ is pronounced "for all $\mathbf{x}$" —intuitively meaning "*for all values of* $\mathbf{x}$" rather than "for all variables $x, y'', z'''_{1234009}, \ldots$

that **x** may stand for".

> We say that the part of A between the two large red brackets above is the scope of $(\forall \mathbf{x})$.
>
> Thus the **x** in $(\forall \mathbf{x})$ and the entire B are in this scope.

□

The "in particular" observation in case (1) along with the cases (2) and (3) make it clear that *every Boolean wff is also a (1st-order) wff.*

> Thus first-order logic can "speak" Boolean (but not the other way around, as we made abundantly clear!)

Oct. 29, 2025

7.1.1 SCOPE of QUANTIFIERS

7.1.9 Example. $x = y$, $\bot$ and p are wff. In fact, Atomic.
The last two are also Boolean wff.

$$\overbrace{((\forall x)\overbrace{((\forall z)(\neg x = y))}^{scope\ of\ (\forall z)})}^{scope\ of\ (\forall x)}$$

y is *free*; an "*input*" variable

Note that $\neg$ in $(\neg x = y)$ *applies to* $x = y$ NOT to x!

Glue cannot apply to an object like x. Must apply to a *statement* (a wff)!

$((\forall y)((\neg x = y) \wedge p))$ and $(((\forall y)(\neg x = y)) \wedge p)$ are also wff.

BTW, in the two last examples: *p is in the scope of $(\forall y)$ in the first*, but not so in the second. □

7.1.2 Existential Quantifier

7.1.10 Definition. (Existential quantifier)

It is convenient —but NOT NECESSARY— to introduce the "*existential quantifier*", $\exists$.

This is only a *metatheoretical* abbreviation symbol that we introduce by this *Definition*, that is, by a "*naming*"

For any wff A, we define $((\exists \mathbf{x})A)$ to be *a short name for*

$$\Big(\neg\Big((\forall \mathbf{x})(\neg A)\Big)\Big) \tag{1}$$

We pronounce $((\exists \mathbf{x})A)$ "for some (value of) $\mathbf{x}$, A holds".

The intuition behind this $((\exists \mathbf{x})A)$ *naming* is captured by the diagram below

$$\Big(\overbrace{\neg}^{\textit{it is not the case that}} \Big(\underbrace{(\forall \mathbf{x})}_{\textit{all values of } x} \overbrace{(\neg A)}^{\textit{make } A \textit{ false}} \Big)\Big)$$

So SOME value of x makes A TRUE.

The *scope of* $(\exists \mathbf{x})$ in

$$\Big((\exists \mathbf{x})A\Big) \tag{2}$$

is the area between the two red brackets.

In particular, the leftmost $\mathbf{x}$ in (2) is in the scope. □

Priorities Revisited

We *augment* our priorities *table*, *from highest to lowest*:

$$\overbrace{\forall, \exists, \neg}^{\text{equal priorities}}, \wedge, \vee, \rightarrow, \equiv$$

Associativities *remain right*! Thus, $\neg(\forall x)\neg A$ is *a short form* of (1) in 7.1.10.

Another example: $(u = v \rightarrow (((\forall x)x = a) \wedge p))$ simplifies into

$$u = v \rightarrow (\forall x)x = a \wedge p$$

More examples:

(2) Instead of $((\forall z)(\neg x = y))$ we write

$$(\forall z)\neg x = y$$

(3) Instead of $((\forall x)((\forall x)x = y))$ we write

$$(\forall x)(\forall x)x = y$$

7.1.3 BOUND vs FREE

.

7.1.11 Definition. A variable **x** *occurs free* in a wff A iff *it is* NOT *inside the scope of a* $(\forall \mathbf{x})$ *or* $(\exists \mathbf{x})$ —otherwise it occurs bound.

We say that a bound variable **x** in $\boxed{(\forall \mathbf{x})A}$ other than the one in the displayed $(\forall \mathbf{x})$, belongs to the displayed leftmost "$(\forall \mathbf{x})$" iff **x** *occurs free in* A —thus

> it was this leftmost "$(\forall \mathbf{x})$", that we added to the left of A, that *did the bounding*!
>
> The terminology "*belongs to*" is now clear.

We apply this criterion to *subformulas* of A of the form $(\forall \mathbf{x})(\ldots)$ to determine where various bound **x** found inside A belong. □

7.1.12 Example. Consider

$$(\forall x)\overbrace{(x = y \to (\forall x)x = z)}^{A}$$

Here the red x in A belongs to the red $\forall x$. The black x belongs to the black $\forall x$. **The occurrences of** y, z **are free.** □

7.1.4 Boolean Abstractions

7.1.13 Remark. We saw that *a Boolean wff, is also a 1st-order wff. We view Boolean formulas as abstractions of 1st-order ones*.

How is this Abstraction (= LESS DETAIL) accomplished?

Well, THEORETICALLY, in any given 1st-order wff we just "hide" all 1st-order features inside certain *glue-less subformulas* that take up *the role of NEW Boolean variables*.

That is, we view *any wff among the following three* forms as Boolean **variables** —none of them has exposed glue

These have no exposed glue so are viewed as Boolean **variables**

1. $t = s$
2. $\phi t_1 t_2 t_3 \dots t_n$
3. $(\forall x)A$

WAIT! I do see "glue" in $(\forall x)(A \to B)$; don't I ???

No, you don't if you are a citizen of Boolean! The "$\to$" is hidden inside the scope of $(\forall x)$.

So an inhabitant of Boolean logic has an INTEREST in the above "Boolean variables" if and only if they are connected with *VISIBLE* Boolean glue to form Boolean wff.

Of course, Boolean logic whose job is to certify tautologies —by either truth tables or proofs— has no use for isolated Boolean variables, that is, ones that are not glued to anything!

Examples.

- In Boolean Country you see this "$x = y \to x = y \vee x = z$" as "$\boxed{x = y} \to \boxed{x = y} \vee \boxed{x = z}$" where the first and second box is the same —say variable p— while the last one is different. You recognize a tautology!

- In fact, In Boolean Country you see this wff

$$A \wedge B \to B \tag{1}$$

as Boolean (it certainly is 1st-order)

Pause. Why "certainly"? ◄

You don't care what A, B are hiding as long as the shape (1) is right.

A secret: In all these abstractions you look for tautologies or tautological implications.

- You see this "$x = x$" as "$\boxed{x = x}$". *Just a Boolean variable*. NOT a tautology.
- The same goes for this "$(\forall x)x = y \rightarrow x = y$" which the Boolean citizen views as "$\boxed{(\forall x)x = y} \rightarrow \boxed{x = y}$", that is, a Boolean wff $p \rightarrow q$. Not a tautology.

Process of abstraction: We only abstract (that is, we see as "Boolean variables") the expressions 1.–3. above in order to turn a 1st-order wff into a Boolean wff.

The three forms above are known in logic as **Prime Formulas**.

Invariably, we abstract in order to detect *tautologies* or *tautological implications* —like $A, B \models_{taut} A \wedge B$— as both are crucial in proofs!

As it will be obvious by observing our several proof examples (to follow), most of the time *it is not necessary* to carry out the abstraction process all the way down to prime formulas.

7.1.5 More Boolean abstraction examples

- If A is

  $p \to x = y \vee (\forall x)\phi x \wedge q$ (note that q *is not* in the scope of $(\forall x)$)

  then we abstract as

  $$p \to \boxed{x = y} \vee \boxed{(\forall x)\phi x} \wedge q \tag{1}$$

  so the Boolean citizen sees

  $$p \to p' \vee p'' \wedge q$$

If we ask "*show ALL the prime formulas in A by boxing them*" then we —*who understand 1st-order language and we* can see inside scopes— would have also boxed ϕx above. The Boolean citizen cannot see ϕx in the scope of $(\forall x)$ anyway so the boxing done by such a citizen would be exactly as we gave it in (1)

- First box <u>all</u> prime formulas in (2) below.

  $$(\forall x)(x = y \to (\forall z)z = a \vee q) \tag{2}$$

  Here it is.

  $$\boxed{(\forall x)(\boxed{x = y} \to \boxed{(\forall z)\boxed{z = a}} \vee q)}$$

  Now abstract the above as if you were a Boolean citizen:

  $$\boxed{(\forall x)(x = y \to (\forall z)z = a \vee q)}$$

  You see no glue at all because you cannot see inside the scope of the leftmost $(\forall x)$*!*

  The abstraction is something like

  "p"

- $x = y \to x = y$ abstracts as $\boxed{x = y} \to \boxed{x = y}$. That is, $p \to p$ or $A \to A$ —*a tautology.*

Consider $\overbrace{(\forall x)x = y}^{A} \wedge \overbrace{(\exists z)z = w}^{B} \to \overbrace{(\forall x)x = y}^{A} \vee \overbrace{(\exists z)z = w}^{B}$. No need to go all the way to prime formulas like $(\forall x)x = y$, etc., to see that this has the form $A \wedge B \to A \vee B$. Hence is a tautology.

Why bother with abstractions? Well, the last example is a tautology so a Boolean citizen can prove it.

However $\boxed{x = x}$ and $\boxed{(\forall x)x = y} \to \boxed{x = y}$ are not tautologies and we need predicate logic techniques to settle their theoremhood.

□ 

We can now define:

7.1.14 Definition. (Tautologies and Tautological Implications)

We say that a (1st-order) wff, A, *is a tautology and write* $\models_{taut} A$ iff *it has a Boolean abstraction that is*.

In 1st-order Logic $\Gamma \models_{\text{taut}} A$ is applied to the Boolean abstraction A and to the abstractions contained in Γ.

So, no matter what A, B, C "say" (via their 1st-order shape) we have, for example,

$$A \equiv B, B \equiv C \models_{taut} A \equiv C$$

□

Definition. A 1st-Order formula Q is a Tautology IFF we found ONE Boolean Abstraction of it that is.

We **do not expect NOR need** that all its abstractions will be tautologies. For example, if we abstract the formula as $A \to B$ then we *did not establish* that it is a tautology.

However if we further note that A is $C \wedge B$, then $A \to B$, aka $C \wedge B \to B$, *IS* recognised as a tautology!

7.1.6 Substitutions

A substitution is a *textual substitution*: **Find and Replace**.

In $A[\mathbf{x} := t]$ we will replace **all** occurrences of a *free* $\mathbf{x}$ in A by the term t: *Find and replace*.

In $A[\mathbf{p} := B]$ we will replace **all** occurrences of a $\mathbf{p}$ in A by B: *Find and replace*.

7.1.15 Example. (What to avoid) Consider the substitution below

$$\Big((\exists x)\neg x = y\Big)[y := x]$$

If we go ahead with it *as a brute force "find and replace" asking no questions*, then we are met by a *serious problem*:

The result

$$(\exists x)\neg x = x \tag{1}$$

says *something DIFFERENT than* what the original formula says!

The original —$((\exists x)\neg x = y)$— says "no matter what the value of y is, there is a *different* x-value".

The above is true in any application of logic *where we have infinitely many objects*. For example, it is true of real numbers and natural numbers.

On the other hand, (1) is *NEVER* true! It says that there is an object that is *different from itself*! □

7.1.16 Definition. (Substitution) Each of

1. In $A[\mathbf{x} := t]$ replace all occurrences of a free $\mathbf{x}$ in A by the term t: *Find and replace*.

2. In $A[\mathbf{p} := B]$ replace all occurrences of a $\mathbf{p}$ in A by B: *Find and replace*.

However we *abort* the substitution 1 or 2 if it so happens that going ahead with it

forces a free variable $\mathbf{y}$ *of* t *or* B *to become bound* because t *or* B *ended up inside the scope of a* $(\forall \mathbf{y})$ *or* $(\exists \mathbf{y})$.

We say that the substitution is undefined in such cases, and that the reason is that *we had a "free variable capture"*.

There is a variant of substution 2, above:

3. In $A[\mathbf{p} \setminus B]$ replace all occurrences of a $\mathbf{p}$ in A by B: *Find and replace*.

For technically justified reasons to be learnt later, *we never abort this one,* **capture or not**.

We call the substitutions 1. and 2. *conditional* or *constrained*, while the substitution 3. unconditional or *unconstrained*.

There is NO unconditional version of 1.

PRIORITIES (AGAIN!)

$[\mathbf{x} := t], [\mathbf{p} := B], [\mathbf{p} \setminus B]$ have higher priority than all connectives $\forall, \exists, \neg, \wedge, \vee, \rightarrow, \equiv$. They associate from LEFT to RIGHT that is $A[\mathbf{x} := t][\mathbf{p} := B]$ means

$$\Big(\big(A[\mathbf{x} := t]\big)[\mathbf{p} := B]\Big)$$

□

7.1.17 Example. Several substitutions based on Definition 7.1.16.

(1) $(y = x)[y := x]$.

The red brackets are META brackets. I need them to show the substitution applies to the whole formula.

The result is $x = x$.

(2) $\Big((\forall x)x = y\Big)[y := x]$. By 7.1.16, this is undefined because if I go ahead then x is captured by $(\forall x)$.

(3) $(\forall x)(x = y)[y := x]$. According to priorities, this means $(\forall x)\Big\{(x = y)[y := x]\Big\}$.
That is, "apply the quantifier $(\forall x)$ to $x = x$", **which is all right**.

Result is $(\forall x)x = x$.

(4) $\Big((\forall x)(\forall y)\phi(x, y)\Big)[y := x]$. This says

- Do $\Big((\forall x)\Big((\forall y)\phi(x, y)\Big)\Big)[y := x]$
- This is all right since y is not free in $\Big((\forall y)\phi(x, y)\Big)$ —so y *not found; no replace!*
  Result is the original formula UNCHANGED.

(5) $\Big(z = a \vee ((\forall x)x = y)\Big)[y := x]$. *Abort*: x is captured when we attempt substitution in the **subformula** $(\forall x)x = y$.

(6) $\Big((\forall x)p\Big)[p \setminus x = y]$ Unconditional substitution. *Just find and replace, no questions asked!*

Result: $(\forall x)x = y$.

(7) $\Big((\forall x)p\Big)[p := x = y]$ Undefined. *x in $x = y$ will get captured if you go ahead!* □

Nov. 3, 2025

7.1.7 Partial Generalisation

7.1.18 Definition. (Partial Generalisation) We say that B is *a partial generalisation* of A if B is formed *by adding as a PREFIX to A* zero or more strings of the form $(\forall \mathbf{x})$ for any choices whatsoever of the variable $\mathbf{x}$ —*repetitions allowed*. □

7.1.19 Example. Here is a small list of partial generalisations of the formula $x = z$:

$x = z$,

$(\forall w)x = z$,

$(\forall x)(\forall x)x = z$,

$(\forall x)(\forall z)x = z$,

$(\forall z)(\forall x)x = z$,

$(\forall z)(\forall y)(\forall z)(\forall x)(\forall u)x = z$. □

7.2 Axioms and Rules for Predicate Logic

7.2.1 Definition. (1st-Order Axioms) These are **all the partial generalisations** of *all the instances of the following schemata*.

1. All tautologies. **Examples**: $x = y \to x = y$ is included here and so are $\top$, $p \to p$ and $r \to p \vee r$ but also ANY WHATSOEVER *instance* of the *SCHEMA* $((A \to B) \to A) \to A$, for ANY specific A and B.
2. $(\forall \mathbf{x})A \to A[\mathbf{x} := t]$

Note that *we get an instance of this schema ONLY IF the substitution is not aborted*.

3. $A \to (\forall \mathbf{x})A$ —*PROVIDED* $\mathbf{x}$ *is not free in* A.
4. $(\forall \mathbf{x})(A \to B) \to (\forall \mathbf{x})A \to (\forall \mathbf{x})B$
5. $\mathbf{x} = \mathbf{x}$
6. $t = s \to (A[\mathbf{x} := t] \equiv A[\mathbf{x} := s])$

Note that *we get an instance of this schema ONLY IF none of the substitutions above is aborted*.

The set of all first-order axioms is named "Λ_1" —"1" for 1st-order.

□

Our **ONLY** INITIAL (or *Primary* or *Primitive*) rule is **Modus Ponens**:

$$\frac{A, A \to B}{B} \qquad (MP)$$

You may think that including all tautologies as axioms is overkill. However

1. It is customary to do so in the literature ([Tou08, Sho67, End72, Man77, Tou03a])
2. After Post's Theorem we do know that every tautology is a theorem of Boolean logic.

Adopting axiom 1. makes every tautology also a theorem of Predicate Logic outright!

This is the easiest way (a literature favourite!) to incorporate Boolean logic as a *sublogic* of 1st-order logic.

7.3 First-order Proofs and Theorems

A Hilbert-style proof from Hypotheses Γ (Γ-proof) is *ALMOST exactly as defined in the case of Boolean Logic.*[†] Namely:

It is a finite sequence of *wff*

$$A_1, A_2, A_3, \ldots, A_i, \ldots, A_n$$

such that each A_i is ONE of

1. Axiom from Λ_1 OR a member of Γ

   OR

2. Is obtained by *MP* from $X \to Y$ and X that appear to the LEFT of A_i (A_i *is the same string as* Y *then.*)

However, here "*wff*" is *1st-order*, and Λ_1 is a DIFFERENT set of axioms than the old Λ. Moreover we have ONLY one rule up in front and it is Neither Leib NOR Eqn!.

As in Boolean definitions, a 1st-order theorem from Γ (Γ-theorem) is a formula that occurs in a 1st-order Γ-proof.

As before we write "$\Gamma \vdash A$" to say "A is a Γ-theorem" and write "$\vdash A$" to say "A is an absolute theorem".

[†]Note the change of start-up (Primitive) Rule, though.

Hilbert proofs in 1st-order logic are written vertically as well, with line numbers and annotation.

The following metatheorems about proofs and theorems

- proof tail removal,
- proof concatenation,
- a wff is a Γ-theorem iff it occurs at the end of a proof
- hypothesis strengthening,
- hypothesis splitting,
- Transitivity of $\vdash$ (4.1.11),

and hence

- usability of derived rules,
- usability of previously proved theorems

hold with the same metaproofs *as in the Boolean case*.

We TRIVIALLY have Post's Theorem (the weak form that we proved for Boolean logic).

7.3.1 Metatheorem. (Weak Post's Theorem for 1st-order logic)

If $A_1, \ldots, A_n \models_{taut} B$ *then* $A_1, \ldots, A_n \vdash B$

Proof. *Exactly the same as in Boolean logic, see 1, p.170*, since the assumption yields $A_1, \ldots, A_{n-1} \models_{taut} A_n \to B$ (*easy!!*) which yields

$$A_1, \ldots, A_{n-2} \models_{taut} A_{n-1} \to A_n \to B$$

etc. etc. etc. and in the end of all this

$$\models_{taut} A_1 \to A_2 \to \cdots \to A_{n-1} \to A_n \to B \tag{2}$$

Thus *(2) is an AXIOM!!!, hence Theorem!!!*

$$\vdash A_1 \to A_2 \to \cdots \to A_{n-1} \to A_n \to B$$

The (red) assumption $A_1, \ldots, A_n$ above and *n applications of MP* chop the A_i —one at a time— from left to right and we are left with a proved B. □

Thus we may use

$$A_1, \ldots, A_n \vdash B$$

as a *DERIVED rule in any 1st-order proof*, *whenever we know that*

$$A_1, \ldots, A_n \models_{taut} B$$

.

THIS IS DONE A LOT IN Hilbert PROOFS!

7.4 Deduction Theorem

This Metatheorem of First-Order Logic says:

7.4.1 Metatheorem. *If* $\Gamma, A \vdash B$*, then also* $\Gamma \vdash A \to B$

OR

7.4.2 Metatheorem. *If I want to prove* $\Gamma \vdash A \to B$ it is enough to prove $\Gamma, A \vdash B$ instead.

WAIT! Did we not already prove this for Boolean Logic? Yes, but to do so we used in an essential way *Boolean Soundness*. See proof of 5.3.1, where we write "By *Boolean Soundness*, etc."

Boolean Semantics will *NOT help* in Predicate Logic, Because 1st-order Semantics are DIFFERENT, and we will present it at the end of the course!

So here we use a direct proof by Induction on the *length* of First-Order Proofs from $\Gamma + A$.*

*Recall that "$\Gamma \cup \{A\}$", "Γ, A" and "$\Gamma + A$" are *alternative notations* for the *same* set of wff!

Proof. We do Induction on the proof length L that we used for $\Gamma, A \vdash B$:

1. Proof of length $L = 1$ (*Basis*). There is only one formula in the proof: The proof must be just

$$B$$

Only three subcases apply:

- $B \in \Gamma$. Then $\Gamma \vdash B$. Hence $\Gamma \vdash A \to B$ by Post (NOTE that $B \models_{taut} A \to B$).
- B *IS* A. So, $A \to B$ is a tautology hence axiom hence $\Gamma \vdash A \to B$.
- $B \in \Lambda_1$. Then $\Gamma \vdash B$. Conclude as in the *first* bullet.

2. *Assume (I.H.) the claim for all proofs of lengths $L \leq n$.*

3. *I.S.*: The proof has length $L = n + 1$:

$$\overbrace{\ldots, B}^{n+1}$$

Let $B \in \Gamma \cup \{A\} \cup \Lambda_1$. Done as in the *Basis*.

Assume instead that it is the result of MP on formulas to the left of B:

$$\overbrace{\underbrace{\ldots, X, \ldots, X \to B}_{\leq n}, \ldots, B}^{n}$$

($\Gamma + A$-proof where $L = n+1$)

By the I.H. the metatheorem is true for proofs of length $L \leq n$. Thus we have

$$\Gamma \vdash A \to X \qquad (*)$$

and

$$\Gamma \vdash A \to (X \to B) \qquad (**)$$

The following Hilbert proof concludes the case and the entire proof:

1)	$A \to X$	$\langle \Gamma$-thm by $(*)\rangle$
2)	$A \to (X \to B)$	$\langle \Gamma$-thm by $(**)\rangle$
3)	$A \to B$	$\langle 1 + 2 +$ Post$\rangle$

The last line proves the metatheorem. □

Nov. 5, 2025

We learn here HOW exactly to handle the quantifier $\forall$.

7.5 Adding (Removing) "$(\forall x)$" to (from) the beginning of a wff.

7.5.1 Metatheorem. (Weak Generalisation) *Suppose that no wff in* Γ has any free occurrences *of* $\mathbf{x}$.

Then if we have $\Gamma \vdash A$, *we will also have* $\Gamma \vdash (\forall \mathbf{x})A$.

> The last line above does **NOT** say that somehow A "implies" $(\forall \mathbf{x})A$.
>
> It rather says that from a proof of A from Γ a proof of $(\forall \mathbf{x})A$ —probably quite different— can be found, *in fact*, constructed.

It is normally the case and of no adverse consequence that A does have free occurrences of $\mathbf{x}$, else $(\forall \mathbf{x})A$ would be trivial.

Proof. Induction on the length L of the Γ-proof used for A.

1. $L = 1$ (*Basis*). There is only one formula in the proof: The proof must be the 1-wff sequence

$$A$$

Only two subcases apply:

- $A \in \Gamma$. Then A has *no free* $\mathbf{x}$, hence $A \to (\forall \mathbf{x})A$ is axiom 3. Thus, we have a Hilbert proof

1)	A	⟨in Γ; hyp⟩
2)	$A \to (\forall \mathbf{x})A$	⟨Axiom 3 —no free $\mathbf{x}$ in A⟩
3)	$(\forall \mathbf{x})A$	⟨1 + 2 +MP⟩

- $A \in \Lambda_1$. Then so is $(\forall \mathbf{x})A \in \Lambda_1$ by partial generalisation:

Note that if axiom A is $(\forall \mathbf{z})(\forall \mathbf{z}') \dots (\forall \mathbf{z_1}) \underbrace{B}_{\text{axiom schema inst.}}$, then $(\forall \mathbf{x})A$ is $(\forall \mathbf{x})(\forall \mathbf{z})(\forall \mathbf{z}') \dots (\forall \mathbf{z_1})B$. Hence an axiom too.

Hence $(\forall \mathbf{x})A$ is in Λ_1, thus $\Gamma \vdash (\forall \mathbf{x})A$ once more. (Definition of Γ-proof.)

AHA! So that's what "partial generalisation" does for us!

2. *Assume (I.H.) the claim for all proofs of lengths $L \leq n$.*

3. *I.S.*: The proof has length $L = n + 1$:

$$\overbrace{\ldots, A}^{n+1}$$

If $A \in \Gamma \cup \Lambda_1$ then we are done by the argument in 1.

Assume instead that A is the result of MP on formulas to the left of it:

$$\overbrace{\underbrace{\ldots, X, \ldots, X \to A}_{\leq n}, \ldots, A}^{n+1}$$
$$\underbrace{\qquad\qquad\qquad\qquad}_{n}$$

By the I.H. we have

$$\Gamma \vdash (\forall \mathbf{x})X \qquad (*)$$

and

$$\Gamma \vdash (\forall \mathbf{x})(X \to A) \qquad (**)$$

The following Hilbert proof concludes this case and the entire proof:

1)	$(\forall \mathbf{x})X$	$\langle \Gamma\text{-thm by } (*) \rangle$
2)	$(\forall \mathbf{x})(X \to A)$	$\langle \Gamma\text{-thm by } (**) \rangle$
3)	$(\forall \mathbf{x})(X \to A) \to (\forall \mathbf{x})X \to (\forall \mathbf{x})A$	$\langle \text{Axiom 4} \rangle$
4)	$(\forall \mathbf{x})X \to (\forall \mathbf{x})A$	$\langle 2 + 3 + \text{MP} \rangle$
5)	$(\forall \mathbf{x})A$	$\langle 1 + 4 + \text{MP} \rangle$

The last line proves the metatheorem. □

7.5.2 Corollary. *If* $\vdash A$, *then* $\vdash (\forall \mathbf{x})A$.

Proof. The condition that no X in Γ has free $\mathbf{x}$ is met: **Vacuously**. Γ is empty! □

7.5.3 Remark.

1. **HOW TO USE Generalisation**: So, the Metatheorem says that if A is a Γ-theorem then so is $(\forall \mathbf{x})A$ as long as the restriction of 7.5.1 is met.

   But then, *since I can invoke* Γ-*THEOREMS* (not only axioms and hypotheses) in a proof, I can insert the Γ-theorem $(\forall \mathbf{x})A$ anywhere AFTER A in any Γ-proof of A where Γ obeys the restriction on $\mathbf{x}$.

   $$\ldots, A, \ldots, \overbrace{(\forall \mathbf{x})A}^{\text{insert at any time after } A}, \ldots$$

2. Why "weak"? Because I need to know how the A was obtained[†] before I may use $(\forall \mathbf{x})A$. □

[†] From an $\mathbf{x}$-less Γ.

7.5.4 Metatheorem. (Specialisation Rule or Spec)

$$\Big((\forall \mathbf{x})A\Big) \vdash A[\mathbf{x} := t]$$

Goes without saying that *IF* the expression $A[\mathbf{x} := t]$ is undefined (*in the event of "capture"*), then we have nothing to prove.

Proof.

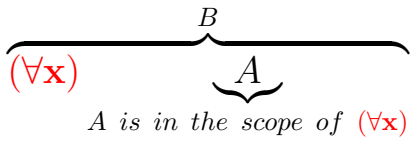
(1)	$(\forall \mathbf{x})A$	$\langle$hyp$\rangle$
(2)	$(\forall \mathbf{x})A \to A[\mathbf{x} := t]$	$\langle$axiom 2$\rangle$
(3)	$A[\mathbf{x} := t]$	$\langle 1 + 2 +$ MP$\rangle$

□

7.5.5 Corollary. $(\forall \mathbf{x})A \vdash A$

Proof. This is the special case where t is $\mathbf{x}$. □

Specialisation removes a (∀**x**) iff the quantifier is the *very first symbol* [a] of a formula B and the entire remaining part of the formula is the scope of that leading (∀**x**):



[a] "(∀**x**)" is ONE compound symbol.

The $(\forall x)$ in the following two CANNOT be removed: $(\forall x)A \vee B$, $A \vee (\forall x)B$.

Really Important! The metatheorems 7.5.5 and 7.5.1 (or 7.5.2) — *which we nickname "spec" and "gen" respectively*— are tools that make our life easy in Hilbert proofs where handling of ∀ is taking place.

7.5.5 *with no restrictions* allows us to REMOVE a leading "(∀**x**)".

Doing so *we might uncover Boolean glue* and thus benefit from applications of "Post" (7.3.1).

If we need to *re-INSERT* (∀**x**) before the end of proof, we employ 7.5.1 to do so.

This is a good recipe for success in 1st-order proofs!

7.5.1 Examples

Ping-Pong proofs.

Hilbert proofs are not well-suited to handle equivalences.

However, trivially

$$A \to B, B \to A \models_{taut} A \equiv B$$

and —by 7.3.1—

$$A \to B, B \to A \vdash A \equiv B \qquad (1)$$

Thus, *to prove* $\Gamma \vdash A \equiv B$ *in Hilbert style* it suffices —by (1), which is a derived rule!— to offer TWO Hilbert proofs:

$$\Gamma \vdash A \to B \text{ AND } \Gamma \vdash B \to A$$

This back and forth motivates the nickname "ping-pong" for this proof technique.

7.5.2 A Few Memorable Examples

7.5.6 Theorem. (Distributivity of ∀ over ∧)
$\vdash (\forall \mathbf{x})(A \wedge B) \equiv (\forall \mathbf{x})A \wedge (\forall \mathbf{x})B$

Proof. By Ping-Pong argument.

We will show TWO things:

1. $\vdash (\forall \mathbf{x})(A \wedge B) \rightarrow (\forall \mathbf{x})A \wedge (\forall \mathbf{x})B$

   and

2. $\vdash (\forall \mathbf{x})A \wedge (\forall \mathbf{x})B \rightarrow (\forall \mathbf{x})(A \wedge B)$

($\rightarrow$) ("1." above)

By DThm, it suffices to prove $(\forall \mathbf{x})(A \wedge B) \vdash (\forall \mathbf{x})A \wedge (\forall \mathbf{x})B$.

(1)	$(\forall \mathbf{x})(A \wedge B)$	⟨hyp⟩
(2)	$A \wedge B$	⟨1 + spec (7.5.5)⟩
(3)	A	⟨2 + Post⟩
(4)	B	⟨2 + Post⟩
(5)	$(\forall \mathbf{x})A$	⟨3 + gen; OK: hyp contains no free $\mathbf{x}$⟩
(6)	$(\forall \mathbf{x})B$	⟨4 + gen; OK: hyp contains no free $\mathbf{x}$⟩
(7)	$(\forall \mathbf{x})A \wedge (\forall \mathbf{x})B$	⟨(5,6) + Post⟩

NOTE. We *ABSOLUTELY MUST* acknowledge for each application of "gen" that *the restriction is met.*

($\leftarrow$) ("2." above)

By DThm, it suffices to prove $(\forall \mathbf{x})A \wedge (\forall \mathbf{x})B \vdash (\forall \mathbf{x})(A \wedge B)$.

(1)	$(\forall \mathbf{x})A \wedge (\forall \mathbf{x})B$	$\langle$hyp$\rangle$
(2)	$(\forall \mathbf{x})A$	$\langle 1 +$ Post$\rangle$
(3)	$(\forall \mathbf{x})B$	$\langle 1 +$ Post$\rangle$
(4)	A	$\langle 2 +$ spec$\rangle$
(5)	B	$\langle 3 +$ spec$\rangle$
(6)	$A \wedge B$	$\langle$(4,5) + Post$\rangle$
(7)	$(\forall \mathbf{x})(A \wedge B)$	$\langle 6 +$ gen; OK: hyp has no free $\mathbf{x}\rangle$

□

Easy and Natural! Right?

7.5.7 Theorem. $\vdash (\forall \mathbf{x})(\forall \mathbf{y})A \equiv (\forall \mathbf{y})(\forall \mathbf{x})A$

Proof. By Ping-Pong. $\vdash (\forall \mathbf{x})(\forall \mathbf{y})A \underset{\leftarrow}{\overset{\rightarrow}{}} (\forall \mathbf{y})(\forall \mathbf{x})A$.

$(\rightarrow)$ direction.

By DThm it suffices to prove $(\forall \mathbf{x})(\forall \mathbf{y})A \vdash (\forall \mathbf{y})(\forall \mathbf{x})A$

(1)	$(\forall \mathbf{x})(\forall \mathbf{y})A$	$\langle$hyp$\rangle$
(2)	$(\forall \mathbf{y})A$	$\langle 1 +$ spec$\rangle$
(3)	A	$\langle 2 +$ spec$\rangle$
(4)	$(\forall \mathbf{x})A$	$\langle 3 +$ gen; OK hyp has no free $\mathbf{x}\rangle$
(5)	$(\forall \mathbf{y})(\forall \mathbf{x})A$	$\langle 4 +$ gen; OK hyp has no free $\mathbf{y}\rangle$

$(\leftarrow)$
Exercise! Justify that you can write the above proof backwards!

□

7.5.8 Example. Say $\underline{A \text{ has no free } x}$. Then $\vdash (\forall x)A \equiv A$. Indeed, $\vdash (\forall x)A \to A$ by ax. 2 and $\vdash A \to (\forall x)A$ by ax. 3.

In particular we also have $(\forall \mathbf{x})\top \equiv \top$, $(\forall \mathbf{x})\bot \equiv \bot$ and $(\forall \mathbf{x})\mathbf{p} \equiv \mathbf{p}$.

□ 

7.5.9 Metatheorem. (Monotonicity of ∀) *If* $\Gamma \vdash A \to B$, *then* $\Gamma \vdash (\forall \mathbf{x})A \to (\forall \mathbf{x})B$, *as long as no wff in* Γ *has a free* $\mathbf{x}$.

Proof.

(1)	$A \to B$	⟨invoking a Γ-thm⟩
(2)	$(\forall \mathbf{x})(A \to B)$	⟨1 + gen; OK no free $\mathbf{x}$ in Γ⟩
(3)	$(\forall \mathbf{x})(A \to B) \to (\forall \mathbf{x})A \to (\forall \mathbf{x})B$	⟨Axiom 4⟩
(4)	$(\forall \mathbf{x})A \to (\forall \mathbf{x})B$	$\langle (2,3) + MP \rangle$

□

We annotate an application of "Monotonicity of ∀" by either writing "A-MON" or writing "∀-MON".

7.5.10 Corollary. *If* $\vdash A \to B$, *then* $\vdash (\forall \mathbf{x})A \to (\forall \mathbf{x})B$.

Proof. Case of $\Gamma = \emptyset$. The restriction is vacuously satisfied. □

7.5.11 Corollary. *If* $\Gamma \vdash A \equiv B$, *then also* $\Gamma \vdash (\forall \mathbf{x})A \equiv (\forall \mathbf{x})B$, as long as Γ *does not contain wff with* $\mathbf{x}$ *free.*

Proof.

(1)	$A \equiv B$	$\langle \Gamma$-theorem$\rangle$
(2)	$A \to B$	$\langle 1 +$ Post$\rangle$
(3)	$B \to A$	$\langle 1 +$ Post$\rangle$
(4)	$(\forall \mathbf{x})A \to (\forall \mathbf{x})B$	$\langle 2 + \forall$-MON (7.5.9)$\rangle$
(5)	$(\forall \mathbf{x})B \to (\forall \mathbf{x})A$	$\langle 3 + \forall$-MON (7.5.9)$\rangle$
(6)	$(\forall \mathbf{x})A \equiv (\forall \mathbf{x})B$	$\langle (4,5) +$ Post$\rangle$

□

7.5.12 Corollary. *If* $\vdash A \equiv B$, *then also* $\vdash (\forall \mathbf{x})A \equiv (\forall \mathbf{x})B$.

Proof. Take $\Gamma = \emptyset$. □

7.6 Weak Leibniz for 1st-Order Logic

Note that since Post's theorem holds in first-order logic, we have that the *two Boolean* primary rules (and all Boolean derived rules; WHY?) hold in predicate logic.

For example, the Boolean Leibniz rule

$$A \equiv B \vdash C[\mathbf{p} := A] \equiv C[\mathbf{p} := B]$$

holds since we have

$$A \equiv B \models_{taut} C[\mathbf{p} := A] \equiv C[\mathbf{p} := B]$$

What makes the rule "Boolean" is that we look at all of A, B, C and $\mathbf{p}$ from the Boolean "citizen's" point of view (Boolean abstractions). In particular, $\mathbf{p}$ is NOT in the scope of any quantifier! Because if IT IS, then the Boolean practitioner CANNOT SEE IT —hence CANNOT USE IT— thus rendering the rule trivial:

$$\frac{A \equiv B}{C \equiv C}$$

Hmmm. Can I do Leibniz with a $\mathbf{p}$ that is IN the scope of a quantifier? You bet!!

7.6.1 Metatheorem. ("Weak" (1st-order) Leibniz "WL")
If $\vdash A \equiv B$, *then also* $\vdash C[\mathbf{p} \setminus A] \equiv C[\mathbf{p} \setminus B]$.

Proof. This generalises 7.5.12.

The metatheorem is proved by *Induction on the (formation of the) wff* C.

Basis. Atomic case for C:

(1) C is $\mathbf{p}$. The metatheorem boils down to "if $\vdash A \equiv B$, then $\vdash A \equiv B$", which trivially holds!

(2) C is *NOT* $\mathbf{p}$ —that is, it is $\mathbf{q}$ (other than $\mathbf{p}$), or is $\bot$ or $\top$, or is $t = s$, or it is $\phi(t_1, \ldots, t_n)$. That is, C does not contain the "text" $\mathbf{p}$.

Then our Metatheorem statement becomes "*if* $\vdash A \equiv B$, *then* $\vdash C \equiv C$".

Given that $\vdash C \equiv C$ is indeed the case by Axiom 1, the "if" part is irrelevant. Done.

The complex cases.

(i) C is $\neg D$. From the I.H. we have $\vdash D[\mathbf{p} \setminus A] \equiv D[\mathbf{p} \setminus B]$,

hence $\vdash \neg D[\mathbf{p} \setminus A] \equiv \neg D[\mathbf{p} \setminus B]$ by Post.

But that is the same as

$$\vdash \overbrace{(\neg D)}^{C}[\mathbf{p} \setminus A] \equiv \overbrace{(\neg D)}^{C}[\mathbf{p} \setminus B]$$

since "$\neg$" is not searched for $\mathbf{p}$.

(ii) C is $D \circ E$, where $\circ \in \{\wedge, \vee, \rightarrow, \equiv\}$.

The I.H. yields $\vdash D[\mathbf{p} \setminus A] \equiv D[\mathbf{p} \setminus B]$ and $\vdash E[\mathbf{p} \setminus A] \equiv E[\mathbf{p} \setminus B]$ hence

$$\vdash \overbrace{D[\mathbf{p} \setminus A]}^{X} \circ \overbrace{E[\mathbf{p} \setminus A]}^{Y} \equiv \overbrace{D[\mathbf{p} \setminus B]}^{X'} \circ \overbrace{E[\mathbf{p} \setminus B]}^{Y'} \text{ by Post.}$$

To see Post relevance, the above follows from this:

Rename:

- $D[\mathbf{p} \setminus A]$: X
- $E[\mathbf{p} \setminus A]$: Y
- $D[\mathbf{p} \setminus B]$: X'
- $E[\mathbf{p} \setminus B]$: Y'

Then $X \equiv X', Y \equiv Y' \models_{taut} X \circ Y \equiv X' \circ Y'$.

This is

$$\vdash \overbrace{(D \circ E)}^{C}[\mathbf{p} \setminus A] \equiv \overbrace{(D \circ E)}^{C}[\mathbf{p} \setminus B]$$

due to the way substitution works, namely,

$(D \circ E)[\mathbf{p} \setminus A]$ is the same wff as $D[\mathbf{p} \setminus A] \circ E[\mathbf{p} \setminus A]$ WHY?

(iii) C is $(\forall \mathbf{x})D$. This is the "*interesting case*".
From the I.H. follows $\vdash D[\mathbf{p} \setminus A] \equiv D[\mathbf{p} \setminus B]$.

From 7.5.12 we get

$$\vdash (\forall \mathbf{x})\Big(D[\mathbf{p} \setminus A]\Big) \equiv (\forall \mathbf{x})\Big(D[\mathbf{p} \setminus B]\Big)$$

also written as

$$\vdash \overbrace{\big((\forall \mathbf{x})D\big)}^{C}[\mathbf{p} \setminus A] \equiv \overbrace{\big((\forall \mathbf{x})D\big)}^{C}[\mathbf{p} \setminus B]$$

because

$$\big((\forall \mathbf{x})D\big)[\mathbf{p} \setminus A] \text{ is the same wff as } (\forall \mathbf{x})\Big(D[\mathbf{p} \setminus A]\Big)$$

□

WL is the only "Leibniz" we will ever *need (practically) in our use of 1st-order logic in these lectures.*

Why "weak"? Because of the restriction on the Rule's Hypothesis: $A \equiv B$ must be an absolute theorem. (Recall that the Boolean Leibniz was not so restricted).

More Memorable Examples and "Techniques".

7.6.2 Theorem. $\vdash (\forall \mathbf{x})(A \to B) \equiv (A \to (\forall \mathbf{x})B)$, as long as $\mathbf{x}$ has no free occurrences in A.

Proof.

Ping-Pong using DThm.

($\to$) I want

$$\vdash (\forall \mathbf{x})(A \to B) \to (A \to (\forall \mathbf{x})B)$$

Better still, let me do (DThm)

$$(\forall \mathbf{x})(A \to B) \vdash A \to (\forall \mathbf{x})B$$

and, even better, (DThm!) I will do

$$(\forall \mathbf{x})(A \to B), A \vdash (\forall \mathbf{x})B$$

(1)	$(\forall \mathbf{x})(A \to B)$	$\langle$hyp$\rangle$
(2)	A	$\langle$hyp$\rangle$
(3)	$A \to B$	$\langle$(1) + spec$\rangle$
(4)	B	$\langle$(2, 3) + MP$\rangle$
(5)	$(\forall \mathbf{x})B$	$\langle$(4) + gen; OK: no free $\mathbf{x}$ in (1) or (2)$\rangle$

($\leftarrow$) I want

$$\vdash (A \to (\forall \mathbf{x})B) \to (\forall \mathbf{x})(A \to B)$$

or better still (DThm)

$$A \to (\forall \mathbf{x})B \vdash (\forall \mathbf{x})(A \to B) \tag{1}$$

Seeing that $A \to (\forall \mathbf{x})B$ has no free $\mathbf{x}$, I can prove the even easier

$$A \to (\forall \mathbf{x})B \vdash A \to B \tag{2}$$

and *after* this proof is done, then I can apply Gen to $A \to B$ to get $(\forall \mathbf{x})(A \to B)$.

OK! By DThm I can prove the even simpler than (2)

$$A \to (\forall \mathbf{x})B, A \vdash B \tag{3}$$

Here it is:

(1)	$A \to (\forall \mathbf{x})B$	$\langle$hyp$\rangle$
(2)	A	$\langle$hyp$\rangle$
(3)	$(\forall \mathbf{x})B$	$\langle$(1, 2) + MP$\rangle$
(4)	B	$\langle$(3) + spec $\rangle$

□

7.6.3 Corollary. *If* $\Gamma \vdash A \to B$ *and* $\mathbf{x}$ *is not free in either* Γ *or* A, *then we have also* $\Gamma \vdash A \to (\forall \mathbf{x})B$.

The operation expressed in the corollary is called "∀-Introduction", in short, "*A-intro*".

Proof. We have $\Gamma \vdash (\forall \mathbf{x})(A \to B)$ by Gen (restriction on Γ makes it OK!). Then viewing 7.6.2 as an Equational proof

$$
\begin{array}{ll}
 & (\forall \mathbf{x})(A \to B) \\
\Leftrightarrow & \langle 7.6.2 \rangle \\
 & A \to (\forall \mathbf{x})B
\end{array}
$$

we have $\Gamma \vdash A \to (\forall \mathbf{x})B$.

BUT ALSO:

1)	$A \to B$	$\langle \Gamma\text{-thm} \rangle$
2)	$(\forall \mathbf{x})(A \to B)$	$\langle 1 +$ Gen; OK, no free $\mathbf{x}$ in $\Gamma \rangle$
3)	$(\forall \mathbf{x})(A \to B) \equiv A \to (\forall \mathbf{x})B$	$\langle 7.6.2 \rangle$
4)	$A \to (\forall \mathbf{x})B$	$\langle 2 + 3 +$Eqn$\rangle$

□

7.6.4 Corollary. $\vdash (\forall \mathbf{x})(A \vee B) \equiv A \vee (\forall \mathbf{x})B$, as long as $\mathbf{x}$ does not occur free in A.

Proof.

$$
\begin{aligned}
& (\forall \mathbf{x})(A \vee B) \\
\Leftrightarrow\ & \langle \text{WL} + \neg\vee \text{ (axiom, so abs. thm!); ``Denom:'' } (\forall \mathbf{x})\mathbf{p} \rangle \\
& (\forall \mathbf{x})(\neg A \to B) \\
\Leftrightarrow\ & \langle \text{``}\forall \to\text{''} \ (7.6.2) \rangle \\
& \neg A \to (\forall \mathbf{x})B \\
\Leftrightarrow\ & \langle \text{tautology, hence axiom} \rangle \\
& A \vee (\forall \mathbf{x})B
\end{aligned}
$$

□

Most of the statements we prove in what follows have *Dual* counterparts obtained by swapping $\forall$ and $\exists$ and $\vee$ and $\wedge$.

Let us give a theorem version of the definition of $\exists$. This is useful in *Equational* proofs in Predicate Logic.

Definition (Recall):

$$(\exists \mathbf{x})A \text{ is short name for } \neg(\forall \mathbf{x})\neg A \tag{1}$$

Next consider the axiom

$$\neg(\forall \mathbf{x})\neg A \equiv \neg(\forall \mathbf{x})\neg A \tag{2}$$

Let me use the *ABBREVIATION* (1) ONLY on *ONE* side of "$\equiv$" in (2). I get the theorem

$$(\exists \mathbf{x})A \equiv \neg(\forall \mathbf{x})\neg A$$

So I can write the theorem without words like this:

$$\vdash (\exists \mathbf{x})A \equiv \neg(\forall \mathbf{x})\neg A \tag{3}$$

HEY! I can apply (3) in Equational proofs —via WL— easily!

I will still refer to (3) in proofs as "Def of E".

Here's something useful AND good practise too! It is the *Dual* of 7.6.4.

7.6.5 Corollary. $\vdash (\exists \mathbf{x})(A \wedge B) \equiv A \wedge (\exists \mathbf{x})B$, as long as $\mathbf{x}$ does not occur free in A.


In annotation we may call the above the "$\exists\wedge$ theorem".


Proof.

$$
\begin{aligned}
& (\exists \mathbf{x})(A \wedge B) \\
\Leftrightarrow\ & \langle \text{Def of E} \rangle \\
& \neg(\forall \mathbf{x})\neg(A \wedge B) \\
\Leftrightarrow\ & \langle \text{WL + axiom 1 (deM); "Denom:" } \neg(\forall \mathbf{x})\mathbf{p} \rangle \\
& \neg(\forall \mathbf{x})(\neg A \vee \neg B) \\
\Leftrightarrow\ & \langle \text{WL + } \forall \text{ over } \vee \text{ (7.6.4) —no free } \mathbf{x} \text{ in } \neg A\text{; "Denom:" } \neg\mathbf{p} \rangle \\
& \neg(\neg A \vee (\forall \mathbf{x})\neg B) \\
\Leftrightarrow\ & \langle \mathbf{Ax1\ (deM)} \rangle \\
& A \wedge \neg(\forall \mathbf{x})\neg B \\
\Leftrightarrow\ & \langle \text{WL + Def of E; "Denom:" } A \wedge \mathbf{p} \rangle \\
& A \wedge (\exists \mathbf{x})B
\end{aligned}
$$

□

Bibliography

[Bar78] Jon Barwise, *An introduction to first-order logic*, Handbook of Mathematical Logic (Jon Barwise, ed.), North-Holland, Amsterdam, 1978, pp. 5–46.

[DS90] Edsger W. Dijkstra and Carel S. Scholten, *Predicate Calculus and Program Semantics*, Springer-Verlag, New York, 1990.

[End72] Herbert B. Enderton, *A Mathematical Introduction to Logic*, Academic Press, New York, 1972.

[GS94] David Gries and Fred B. Schneider, *A Logical Approach to Discrete Math*, Springer-Verlag, New York, 1994.

[Jec78] T. J. Jech, *Set theory*, Academic Press, New York, 1978.

[Lev79] A. Levy, *Basic Set Theory*, Springer-Verlag, New York, 1979.

[Man77] Yu. I. Manin, *A Course in Mathematical Logic*, Springer-Verlag, New York, 1977.

[Sho67] Joseph R. Shoenfield, *Mathematical Logic*, Addison-Wesley, Reading, MA, 1967.

[Tou03a] G. Tourlakis, *Lectures in Logic and Set Theory, Volume 1: Mathematical Logic*, Cambridge University Press, Cambridge, 2003.

[Tou03b] ———, *Lectures in Logic and Set Theory, Volume 2: Set Theory*, Cambridge University Press, Cambridge, 2003.

[Tou08] ———, *Mathematical Logic*, John Wiley & Sons, Hoboken, NJ, 2008.