Alexander Kasiukov

Suffolk County Community College
MAT 101
Mathematical Reasoning

Logic — The Boring Chapter 0

August 25, 2023

Contents

1 Preface

1.1 The Purpose of these Notes

These notes were written as a supplement for an introductory course MAT101: *A Survey of Mathematical Reasoning* taught by the author in Fall 2022 at Suffolk County Community College. The bulk of the content was completed in July 2022.

1.2 What is Logic?

`Logic` is a discipline of human mind, instructing us how to build

- the vessels that give the proper form to our thoughts, and
- the conduits directing our reasoning towards the truth.

The meaning of those words, "thought", "truth", ... — will be made more precise in what follows, thus giving the word "logic" more specific content as well. However, that content, while far more practical than the above description, will also be transient and incomplete — and thus misleading for someone not inclined to assume a higher vantage point. Logic-that-we-know-now is a result of the historical developments that gave us only an approximation of the true essence of the subject. Logic is both a theoretical and an experimental field, continually growing from the tension between its foundations and applications. That ongoing progress precludes us from taking the current state of logic as the ultimate limit of its ability, and the full scope of its meaning. Let the general goal of finding the truth guide us in our studies, even if we fall short of that ideal.

1.3 What to Expect from this Course

The dualism, of logic-as-an-object of study versus logic-as-a-tool for studying other things, should be a part of any introduction into the subject. Indeed, no tool can be effectively used without at least some basic understanding of its mechanism, and no theory is meaningful until it is put to practice. Thus these notes will have two — intertwined but distinct — narratives: an instruction on how to build those vessels and conduits for human thoughts, and a demonstration on how the tools we build can facilitate our thinking.

For clarity, we will use the word `meta-logic` for the narrative describing logic-as-an-object of study.

Since it is logic itself that gives us the ability to be precise, the meta-logical part of a basic introduction into the subject must inadvertently be informal, intuitive and — as mathematicians often put it — "hand-waiving". Indeed, without becoming circular, such an introduction cannot employ the tools that would allow it to be rigorous. Thus, don't expect anything else from the these notes: when introducing logical concepts, we will aim at achieving *intuitive* clarity without the pretense of rigor and precision[1]. However, as we progress through the material, we will pick the tools along the way that will enable us to be more and more exacting in our discourse.

Finally, in a more advanced course, you will be able to apply the tools we develop here to the study of logic itself, making the study-of-logic-with-logic look like the uroboros symbol:

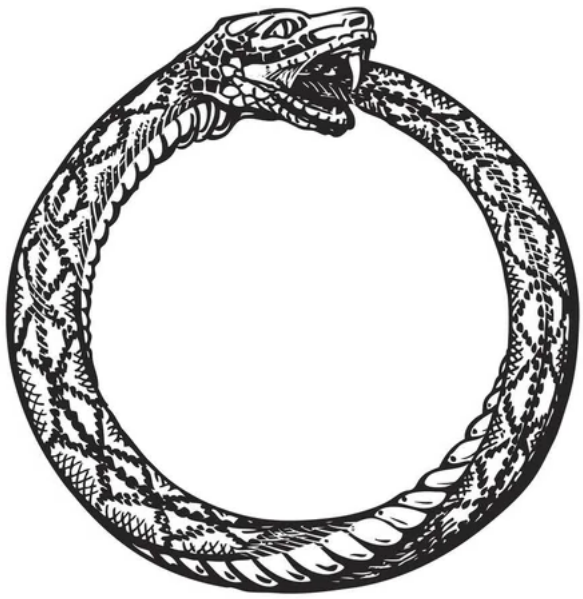


Figure 1: The Uroboros

[1]This introduction should be taken in that spirit as well.

1.4 The Phenomenon of Self-reference

The uroboros captures the idea of self-reference which is perhaps the most intriguing aspect of logic. No story of logic is complete without this topic.

From the first appearance of liar's paradox attributed to Epimenides[2] to the modern work in the foundations of mathematics, self-reference and the resulting paradoxes feed the drama of logical development[3] and make the discipline of logic as interesting as it is. Without the paradoxes causing periodic catastrophes and reshaping of the whole discipline, logic would be perhaps a bit more deep, but no more exciting than a washing machine owner's manual. More importantly, the world we live in would be much more regular and mechanical, and thus less humane. The paradoxes of logic capture the aspect of our existence that elevates the imperfect human mind to the level on which the universe itself operates.

Sadly, what we will have time for in this course is only the introductory basic part of logic that lays the foundations for the study of self-reference without following through on this promise. Thus it can be properly termed "the boring part" of the subject. I hope the glimpses and the shadows of the uroboros you see through these notes — if only superficially — will inspire you to go further in your studies.

[2] Ἐπιμενίδης [Epimenides of Crete] was a semi-mythical Greek philosopher-poet who supposedly lived sometime around 7th or 6th century BC. The paradox stems from a poem Κρητικά [Cretica], attributed to him and quoted in the New Testament. There is no evidence Epimenides himself considered the verse of that poem, "Cretans, always liars", paradoxical. The paradox can be rephrased as follows. Epimenides says: "All Cretans always lie". But Epimenides is a Cretan himself. Is his statement true or false?

[3] More on that in the section on the history of this discipline.

2 Propositional Logic

2.1 Ontology: Statements and Logical Gates

`Ontology` describes the *notions* used in a field of study. The first notion of logic which is used to structure its field of study is that of a statement.

Definition (of `statement`): A `statement` expresses the general idea of such everyday concepts as fact, judgment, sentence, claim etc. The most important property of a statement is its `truth` or `falsehood`, referred to as its `truth value`. When determining whether something is or is not a statement, we are not concerned with deciding its truth value. We just need to make sure that the truth value is a property that makes sense when applied to the thing we are considering. ♢

Example (statement): This dog is big. ♢

Example (non-statements):
Is that dog big?
green
Give me that dog! ♢

The above examples also show one important aspect of statements: statements acquire their full meaning from their specific context. We will pay close attention to the subject of context later.

Statements can be constructed from other statements in various ways.

Example (combining statements): Two statements, "This dog is big." and "You should take it outside." can be combined into one: "This dog is big and you should take it outside." ♢

Definition (of `logical gate`): A `logical gate` is a *particular way* of constructing a new statement from one or more other statements, independent from the specifics of the statements being used in the construction. Each gate is defined by the truth value of the resulting statement for each possible combination of truth values of the original statements. It is convenient to represent that information in the form of so called "truth tables". ♢

For instance, the "combining statements" example on page 6 used the *conjunction* of two statements in question.

Definition (of `logical gate ‘‘conjunction’’`): The `conjunction` of two statements A and B, denoted $A \wedge B$, is a statement whose truth value is defined by the following table:

A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

In natural language, conjunction is usually expressed by the word "and". For example, "this dog is big *and* friendly" is logically the same as

"(this dog is big) $\wedge$ (this dog is friendly)".

However, many different constructions have the same *logical* meaning, with their differences expressing additional (non-logical) meaning variations, as in "this dog is big *but* friendly". $\diamondsuit$

Example (combining statements using specific gate): Using conjunction, we can rewrite the "combining statements" example on page 6 as:

(This dog is big and you should take it outside.) =
(This dog is big.) $\wedge$ (You should take it outside.)

$\diamondsuit$

Definition (of `logical gate ‘‘negation’’`): The negation of a statement A, denoted $\neg A$ is a statement whose truth value is defined by the following table:

A	$\neg A$
T	F
F	T

$\diamondsuit$

Since the list of combinations of truth values of two statements does not depend on the logical gate being considered, we can combine several truth

tables into one, with that list occupying the first two columns, and each new logical gate represented by every subsequent column. This is done in the following table, where we define a few more standard logical gates. The labels "Operation Title" and "Possible Meaning" refer to their rows, rather than the first column.

Definition (of `disjunction, XOR, implication, equivalence`):

Operation Title:		disjunction	XOR	implication	equivalence
Possible Meaning:		or	either... or	if... then	if and only if
A	B	$A \vee B$	$A \veebar B$	$A \Rightarrow B$	$A \Leftrightarrow B$
T	T	T	F	T	T
T	F	T	T	F	F
F	T	T	T	T	F
F	F	F	F	T	T

XOR stands for "exclusive OR". $\diamondsuit$

Even though there are other logical gates which are even occasionally useful in modeling natural language, these are the main logical gates of propositional logic.

HOMEWORK: How many logical gates combining two statements are there in the total?

Importantly, negation, conjunction and disjunction are sufficient for constructing all other logical gates, no matter the number of their constituent statements. We will study that question later.

2.1.1 On the Special Role of Implication

Accepting certain statements as true may require acceptance of certain other statements as true, purely because of the *structure* of those statements. This idea[4] — the idea of formal `inference` — is the very foundation of logic itself. Implication is the formal model of the inference relation between statements. Thus, the concept of implication is the center of logic.

[4]Explicitly stated already by Aristotle, and possibly understood even earlier.

Example (inference): Believing that a black cat is hidden in a particular box necessitates believing that a cat (of any color) is in the box. Using implication, we can express this idea as

$$(\text{a black cat is in the box}) \Rightarrow (\text{a cat is in the box}).$$

♢

Definition (of `argument`): An `argument` is a statement which has the form of an implication. In an argument $A \Rightarrow B$, statement A can be called the `assumption`, the `hypothesis`, the `premise`, or the `antecedent` of the argument; statement B can be called the `consequence`, the `conclusion`, or the `consequent` of the argument. ♢

Definition (of `validity`): An argument is `valid` if and only if it is a true statement, as prescribed by the truth table that defines implication (see page 8).

As we can read from the truth table of implication on page 8, the statement A implies the statement B if and only if whenever A is true, the B must be true as well. (To put it differently, it must be impossible for A to be true and for B to be false.) Thus, an argument is valid if and only if in any circumstances when the assumption A is true, the conclusion B is true as well. ♢

Now we can formulate the notion of logic in more precise terms. Logic is the technology for

- giving our thoughts the right structural form;
- determining their truth value when it can be done based on their structure alone; and
- using the structure of our hypotheses[5] for making valid inferences.

This falls short of our ultimate goal — that of finding the truth. Logic — at least in its present form — guarantees neither the truth nor the meaningfulness of what it offers. It is merely the *grammar* of rational thinking that

[5] i.e. the statements that we are willing to accept, if only provisionally

gives our thoughts the structure making them unambiguous and precise, and provides the formal rules of inference. It is the meter and rhyme of rational thinking, enabling it to express the poetry of truth. Logic by itself cannot *give* us the truth, it can merely help us *formulate* our beliefs and *explicate* the truth already contained in our assumptions. We need to venture outside of logic to come up with reasonable assumptions. Logical conclusions are formal and relative — relative to the truth of our assumptions. Absolute truth of the conclusion is the focus of the following concept:

Definition (of `soundness`): An argument is `sound` if and only if it is valid and starts with a true hypothesis. $\diamondsuit$

Definition (of `modus ponens`): Modus ponens (Latin for "method of affirming") is a fundamental principle of logic affirming the truth of the conclusion of a sound argument:

$$\Big(\big(A \Rightarrow C \big) \wedge A \Big) \Rightarrow C.$$

If the assumption A implies the conclusion C (meaning that the argument $A \Rightarrow C$ is valid) and the assumption A is true (meaning, together with the previous, that the argument is sound), then the conclusion C is true. $\diamondsuit$

Before we move on to the next section, let's introduce some additional terminology related to implications.

Definition (of `converse, inverse, counter-positive`): Suppose we have a statement in the form of an implication

$$A \Rightarrow B$$

where A and B are some other statements. Then

- statement $B \Rightarrow A$ is called "the `converse`" of the original;
- statement $(\neg A) \Rightarrow (\neg B)$ is called "the `inverse`" of the original;

- statement $(\neg B) \Rightarrow (\neg A)$ is called "the `counter-positive`" of the original.

♢

HOMEWORK: Use the truth tables on pages 7, 8 (that define negation, implication and equivalence) to verify the equivalence of

- the original statement and its counter-positive;
- the converse and the inverse.

2.2 Apologia: Truth Tables

`Apologia` is the formal defense of certain position, conduct or actor. In these notes, we will use this term to describe how arguments are *validated* in a particular logical theory.

Propositional logic gives us the tools for only the most basic analysis of reasoning. Such analysis can only give the lowest resolution picture, based on breaking down real life narrative into statements and logical gates. The smallest units in this breakdown process, besides the gates, are the so-called `atomic` statements, namely those that cannot be represented as combinations of smaller statements[6]. Apologia of propositional logic will be based on propositional analysis, namely breaking an argument into atomic statements and gates and applying the truth tables to the resulting implication formula.

Example (valid argument): This dog is always happy after a meal. However, it seems to be troubled and restless. Probably it is hungry. ♢

Example (invalid argument): This dog is always happy after a meal. If you don't feed it, it will be very angry. ♢

What makes one argument valid and another one invalid? How can we effectively decide these questions in general? This is the subject of this

[6]Even though in the later sections, those "atomic" statements will be — sometimes — broken further into smaller pieces of information, those pieces will not be statements.

section.

The first step in determining validity of an argument is its *propositional analysis*, namely breaking the whole of the argument as a statement into smaller statements combined together by logical gates.

In natural communications, we rely on the context when omitting implicit assumptions, and use the flexibility of our language to convey shades of meaning and to avoid rigid repetitiveness. These features make our conversations more succinct and lively, but obscure the structure. To "correct" these shortcomings, we need to make all omitted assumptions explicit and adjust the wording — without change in meaning — in order to reveal the argument's ultimate structure. Some authors even introduce additional terminology to stress the latter point, using the term `proposition` to describe the underlying meaning that may be expressed in various ways by different statements.

Take the first example, "This dog is always happy after a meal. However, it seems to be troubled and restless. Probably it is hungry." Consider the whole argument as one statement. First, we can break that statement along the boundaries of the *sentences*, explicating the logical gates and the grouping that holds it together. At the cost of adding some redundancy, we will also make the individual sentences a bit more self-sufficient, so that one sentence does not depend on the context introduced by another. Some (non-logical) shades of meaning will be lost in this analysis.

$$\Big(\text{(this dog is always happy after a meal)}\wedge \text{(this dog is troubled and restless)}\Big) \Rightarrow \text{(this dog is hungry)}.$$

If we replace different statements with different letters, we get

$$(A \wedge B) \Rightarrow C,$$

which cannot possibly be a valid argument. Indeed, take A and B true and C false. This choice of the truth values will make the whole implication false demonstrating that this argument is invalid.

On the other hand, it should be intuitively clear that what we had before this replacement of statements with letters was a valid argument. How can we reconcile these two conclusions?

The problem here is the insufficient depth of our analysis. To demonstrate validity of this argument, we need to break it down further to reveal more of its propositional structure:

$$\Bigg(\Big((\text{the dog has eaten}) \Rightarrow (\text{the dog is happy}) \Big) \wedge \Big(\neg\, (\text{the dog is happy}) \Big) \Bigg) \Rightarrow \Big(\neg\, (\text{the dog has eaten}) \Big)$$

This looks pretty cumbersome. To make it a bit more readable, a different notation is usually favored in this situation:

$$\frac{\begin{array}{l}(\text{the dog has eaten}) \Rightarrow (\text{the dog is happy}) \\ \neg\, (\text{the dog is happy})\end{array}}{\neg\, (\text{the dog has eaten})}$$

In the above, the horizontal line stands for the main implication of the argument and can be read as "therefore". The assumption of the argument is above the horizontal line, and the conclusion is below that line. The assumption is broken — as much as possible — into a conjunction of smaller statements, written individually one per line. These conjunctions are implicit in this form of writing.

To focus on the statement-gate structure of this argument, substitute the atomic statement "the dog has eaten" with E, and "the dog is happy" with H. Then it becomes:

$$\frac{\begin{array}{l}E \Rightarrow H \\ \neg H\end{array}}{\neg E}$$

or, going back to the original form (which is less cumbersome now):

$$\Bigg(\Big(E \Rightarrow H \Big) \wedge \Big(\neg H \Big) \Bigg) \Rightarrow \Big(\neg E \Big)\,,$$

Consider the truth table of this formula. To complete it, we used the truth tables around page 8 defining implication, conjunction and negation:

E	H	$\Big(\Big(E \Rightarrow H\Big) \wedge \Big(\neg H\Big)\Big) \Rightarrow \Big(\neg E\Big)$
T	T	T
T	F	T
F	T	T
F	F	T

Its last column shows that the argument in question is true for every possible combination of the truth values of the ingredient statements E and H. This is exactly the indicator we have been looking for.

Definition (of `tautology`): A propositional formula is called a `tautology` if and only if it is true for any combination of truth values of its ingredient statements. $\diamondsuit$

Theorem (*Valid Propositional Argument is a Tautology*). *An argument of propositional logic is valid if and only if that argument is a tautology.* $\diamondsuit$

Proof. [7] For an argument, being a tautology means having true conclusion whenever the assumptions of the argument are true. This is exactly the definition of validity of an argument. $\square$

The argument whose validity we verified in this example is called `modus tollens`, which is Latin for "method of removing" — meaning removing the assumption whenever it leads to a false conclusion.

HOMEWORK: Verify, using the truth tables of implication and conjunction, that the logical formula expressing modus ponens (on page 10) is a tautology. Thus modus ponens itself is a valid argument.

[7] Right now, we use the word "proof" informally, as a substitute for "a (hopefully) convincing explanation". The concept of proof will be the center of our attention later, when we will give it a precise definition.

***

Let's analyze the second argument "This dog is always happy after a meal. If you don't feed it, it will be very angry." — the same way we did the first one. Explicating the logical gates and rephrasing some parts of the original argument to match the instances of the same atomic statement across different sentences, we get:

$$\Big((\text{the dog has eaten}) \Rightarrow (\text{the dog is happy})\Big) \Rightarrow \Bigg(\Big(\neg\,(\text{the dog has eaten})\Big) \Rightarrow \Big(\neg\,(\text{the dog is happy})\Big)\Bigg),$$

or, in a more concise form:

$$\frac{(\text{the dog has eaten}) \Rightarrow (\text{the dog is happy})}{\neg\,(\text{the dog has eaten}) \Rightarrow \neg\,(\text{the dog is happy})}$$

Using the same abbreviations as before, the statement-gate structure of this argument can be written as:

$$\frac{E \Rightarrow H}{(\neg E) \Rightarrow (\neg H)}$$

or, in the formula form:

$$\Big(E \Rightarrow H\Big) \Rightarrow \Bigg(\Big(\neg E\Big) \Rightarrow \Big(\neg H\Big)\Bigg).$$

The truth table of this formula can be computed like before:

E	H	$\Big(E \Rightarrow H\Big) \Rightarrow \Bigg(\Big(\neg E\Big) \Rightarrow \Big(\neg H\Big)\Bigg)$
T	T	T
T	F	T
F	T	F
F	F	T

The F in the last column signals that the argument is invalid. It corresponds to the case when E is false and H is true. In that case, the assumption of the argument, namely $E \Rightarrow H$, is true, but the conclusion $\Big(\neg E\Big) \Rightarrow \Big(\neg H\Big)$ is false.

Definition (of `counterexample`): For any given argument, a situation making its assumptions true and conclusion false is called a `counterexample` to that argument. For instance, the combination of false E and true H is a counterexample to the argument we are considering. $\diamondsuit$

A counterexample to an argument shows that the conclusion of that argument is not supported by the assumption. Thus, an argument is invalid if and only if it has at least one counterexample.

Note that an invalid argument can full well have a true conclusion. Validity of an argument has nothing to do with the truth or falsehood of that argument's assumption or conclusion. It merely concerns itself with whether or not the conclusion is supported by the assumption.

Furthermore, within logical discourse, we are not concerned whether our counterexamples are feasible in the real world. Any combination of truth values of the constituent atomic statements can serve as a (counter)example. If the counterexample is indeed impossible in the real world, it means that our assumptions don't capture full relevant details of the situation we want to model in our argument. When you encounter an intuitively correct argument with true conclusion that is formally incorrect, most likely there is a problem with the assumptions not providing an accurate model of the world, or the analysis of the argument not going deep enough to reveal the structure that makes the argument valid.

2.3 Diversion: Backus Notation

Precise way of speaking about any subject requires a *language* capable of carrying the intended meaning with the needed level of precision. A language is defined by its alphabet and expressions, and may be described using symbols and grammars.

Definition (of `alphabet and letters`): The `letters` of a language are the smallest building blocks of its expressions. Taken together, all the letters form the `alphabet` of the language. Formally speaking, an alphabet is any *finite* set. ♢

Definition (of `language`): A `formal language` is a specific (and usually infinite) set of finite sequences of letters from a fixed alphabet. Each of those sequences is called an `expression` of the said language. ♢

Definition (of `grammar`): A grammar of a language is a set of rules permitting to decide effectively, for any finite sequence of letters, whether or not that sequence is an expression of the language. ♢

Definition (of `symbols`): `Symbols` of a language are the building blocks of its expressions. In a way, symbols are between the letters and the expressions. Formally speaking, symbols are finite sequences of letters more general than the expressions. Symbols help us define grammars (and are specific to grammars rather than to the language defined by those grammars). ♢

Example (Backus notation for describing comma-separated lists): Suppose we want to give a precise description for "a list of one or more digits, separated by commas followed by space"[8]. We can describe the grammar of the language of such lists using what is called `Backus Notation`[9]. This notation defines the grammar of a language incrementally, by listing the `production rules` for its symbols.

[8] One may ask why would we need a separate notation for what we just described using natural language. It turns out that in a more complicated situation — for example, when specifying a programming language — the notation we are about to introduce would be much more precise and succinct, which makes it worth the effort.

[9] It is often called `Backus-Naur Form` or `Backus Normal Form`. Both of these terms are incorrect and misleading, so we settle for `Backus Notation` instead of the more standard terminology.

```
<list> ::= <digit>|<digit>, <list>

<digit> ::= 0|1|2|3|4|5|6|7|8|9
```

In this description,

- the symbol[10] "`::=`" means "can be replaced with"; each line with this symbol defines a new `production rule`.
- the words or phrases between angle brackets, like `<digit>`, are called `non-terminal` symbols. Different grammars using different non-terminals may end up describing the same language. The non-terminals pertain not to the language itself, but to a particular grammar. Non-terminals are used to classify various fragments of the expressions and help structure the grammar. The non-terminal symbols of this particular grammar are `<list>` and `<digit>`.
- the symbol `<list>` is the so-called `start symbol` of the grammar, defining the name of a well-formed *expression* of this language. It is a particular special type of a non-terminal symbol. Every grammar should have exactly one start symbol.
- the words and phrases written without any quotation marks or angle brackets are called `terminal` symbols of the language; they are used verbatim to construct the expressions of the language. The terminal symbols of our language are the ten digits, the comma and the space.
- the symbol[11] "|" means "or". In principle, we could go without it, using several production rules in place of one, as in:

```
<list> ::= <digit>
<list> ::= <digit>, <list>
<digit> ::= 0
<digit> ::= 1
<digit> ::= 2
...
```

[10] Symbol of the Backus notation itself, not of the language we are defining.
[11] Again, this is a symbol of the Backus notation, not the language we are defining.

♢

In all the grammars we will consider in these notes, the space symbol will not convey any meaning, permitting us to add it liberally to our expressions just to make them more readable or to stress something in particular. From the formal point of view, all spaces in such situations should be ignored as if they were not there.

How does a grammar help us decide if a sequence of letters constitutes an expression of the language defined by that grammar? The following definition introduces the concept answering this question.

Definition (of `parse tree`): When a language is defined by a grammar, each expression of that language must come from a tree. The root of that tree must be the start symbol of the grammar, each node must correspond to a particular production rule, and each leaf must be a terminal symbol. All leafs taken together should give the expression itself. Such a tree is called the `parse tree` of the expression. ♢

Example (parse tree):

For example, the list "1, 5, 7" has the parse tree

```
<list> ::=
           <digit> ::=
                       "1"
           ", "
           " "
           <list>  ::=
                       <digit> ::=
                                   "5"
                       ","
                       " "
                       <list>  ::=
                                   <digit> ::=
                                               "7"
```

♢

HOMEWORK: Use the Backus notation to specify the grammar for all possible (signed or unsigned) decimals.

The parse tree of an expression gives the precise meaning to that expression. Thus each comprehension task involves parsing stage.

2.4 Standard Identities of Propositional Logic

There are many identities connecting different propositional formulas, similar in kind to the familiar associativity, commutativity and other identities of arithmetic. We already eluded to the fact that the implication $A \Rightarrow C$ is equivalent to stating that either the assumption A is false and we are "off the hook" for any conclusions we may make, or the conclusion C must be true:

$$\Big(A \Rightarrow C\Big) \quad \Leftrightarrow \quad \Big((\neg A) \vee C\Big) \ .$$

Also worth mentioning is the fact that the equivalence of two statements means that each of these statements implies the other:

$$\Big(A \Leftrightarrow B\Big) \quad \Leftrightarrow \quad \Big((A \Rightarrow B) \wedge (B \Rightarrow A)\Big) \ .$$

There is a lot of similarity between addition and multiplication on one hand, and conjunction and disjunction — on the other. Conjunction and disjunction share the properties of commutativity and associativity with the two arithmetic operations.

HOMEWORK: Formulate and verify using the truth tables the properties of commutativity and associativity for disjunction and conjunction.

A more subtle point is the existence of a neutral element. If we consider logical gates as operations on *truth values* of the constituent statements, rather than the statements themselves, then conjunction and disjunction share the property of a neutral element — in this case neutral truth value — with the arithmetic operations. Recall, that a neutral element n of a binary operation $\bullet$ is the element with the property $n \bullet x = x \bullet n = x$ for any x that can be used with that operation. For example, the neutral element of addition is zero, because $0 + x = x + 0 = x$ for any number x.

HOMEWORK: Which one of the two truth values, "true" and "false", is the neutral one for disjunction? Which one is neutral element for conjunction?

2.4.1 Disjunctive Normal Form

It turns out that any logical gate (no matter how many statements it combines) can be expressed as a disjunction of elementary conjunctions. More precisely, disjunctive normal form is defined by the following Backus notation.

Definition (of `disjunctive normal form`):

```
<DNF> ::=
    <Elementary Conjunction> | <Elementary Conjunction> ∨ <DNF>

<Elementary Conjunction> ::=
    <Term> | ( <Term> ∧ <Elementary Conjunction> )

<Term> ::= <Variable> | ¬ <Variable>

<Variable> ::= A | B | ...
```

♢

(Notice also that we started to use spaces for readability.)

HOMEWORK: What are the start symbol, the terminal symbols, the non-terminal symbols of this grammar?

Example (disjunctive normal form):

$$\Big(A \wedge B\Big) \vee \Big((\neg A) \wedge (\neg B)\Big)$$

is a disjunctive normal form. $\diamondsuit$

HOMEWORK: Construct the parse tree of the disjunctive normal form for the above DNF.

Given the truth table of a logical gate, it is extremely easy to determine its DNF. One just needs to write a term for each row with T output listing all inputs equal to T as themselves, and all inputs equal to F as their negations.

Example (DNF of implication): Take the implication

A	B	$A \Rightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

The first line gives the elementary conjunction $A \wedge B$; the second line results in F and thus does not give any elementary conjunction; the third gives $(\neg A) \wedge B$, and the fourth results in $(\neg A) \wedge (\neg B)$. Combining all these elementary conjunctions together, we get the disjunctive normal form of implication:

$$\Big(A \Rightarrow B\Big) \Leftrightarrow \Big(\Big(A \wedge B\Big) \vee \Big((\neg A) \wedge B\Big) \vee \Big((\neg A) \wedge (\neg B)\Big)\Big) .$$

$\diamondsuit$

Similar to the identities of arithmetic, logical operations of conjunction, disjunction and negation satisfy certain identities. Some of these identities are discussed in the following two sections.

2.4.2 De Morgan Laws

Theorem (*negation of disjunction*). *Negation of a disjunction is the conjunction of individual negations:*

$$\neg(A \vee B \ldots) \quad \Leftrightarrow \quad (\neg A) \wedge (\neg B) \ldots \ .$$

♢

Theorem (*negation of conjunction*). *Negation of a conjunction is the disjunction of individual negations:*

$$\neg(A \wedge B \ldots) \quad \Leftrightarrow \quad (\neg A) \vee (\neg B) \ldots \ .$$

♢

2.4.3 Distributive Laws

Theorem (*distributivity of conjunction with respect to disjunction*).

$$A \wedge (B \vee C \ldots) \quad \Leftrightarrow \quad (A \wedge B) \vee (A \wedge C) \ldots \ .$$

♢

Theorem (*distributivity of disjunction with respect to conjunction*).

$$A \vee (B \wedge C \ldots) \quad \Leftrightarrow \quad (A \vee B) \wedge (A \vee C) \ldots \ .$$

♢

HOMEWORK: Verify these four theorems using truth tables.

Example (using logical gates and identities in solving equations): Suppose we want to solve the equation

$$\frac{x^2-4}{x-2}=2.$$

One possible way to go about it is to find common denominator and get everything on one side:

$$\frac{x^2-4}{x-2}=2 \quad \Leftrightarrow \quad \frac{x^2-4}{x-2}=\frac{2x-4}{x-2} \quad \Leftrightarrow$$
$$\frac{x^2-4-2x+4}{x-2}=0 \quad \Leftrightarrow \quad \frac{x^2-2x}{x-2}=0 \quad \Leftrightarrow$$
$$\frac{x(x-2)}{x-2}=0.$$

Since a fraction is zero if and only if the numerator is, and the denominator isn't zero, the last equation is equivalent to the system:

$$\begin{cases} x(x-2)=0 \\ x-2\neq 0. \end{cases}$$

This simultaneous system is just a conjunction of two statements written in a different form. The first equation states that a product is zero. That means that one of the factors is zero. Thus

$$\begin{cases} x(x-2)=0 \\ x-2\neq 0 \end{cases} \quad \Leftrightarrow \quad \begin{cases} \left[\begin{array}{l} x=0 \\ x-2=0 \end{array}\right. \\ x-2\neq 0. \end{cases}$$

where the square bracket is just another way of expressing disjunction.

Now we can use the distributivity of conjunction with respect to disjunction which then leads to the solution in one step:

$$\begin{cases} \left[\begin{array}{l} x=0 \\ x-2=0 \end{array}\right. \\ x-2\neq 0 \end{cases} \quad \Leftrightarrow \quad \left[\begin{array}{l} \begin{cases} x=0 \\ x-2\neq 0 \end{cases} \\ \\ \begin{cases} x-2=0 \\ x-2\neq 0 \end{cases} \end{array}\right. \quad \Leftrightarrow \quad x=0.$$

$\diamondsuit$

This example illustrates one particularly good way of presenting a solution of an equation, inequality, or a system thereof. It is called the `method of equivalence transformations`. In the context of this method, equivalence means *preservation of the the solution set* as we move from one step to the next. In our specific example, it means that the original equation

$$\frac{x^2-4}{x-2}=2$$

has the same solutions as the (trivial) equation $x = 0$, meaning that 0 is the solution of the original equation. When using this notation, one warning about order of operations is necessary.

In arithmetic, we usually drop parentheses when the same operation is repeated over and over. It can be done without harm in $x + y + z$ because of associativity of addition, which makes grouping insignificant. We also do it in non-associative situations, like $x \div y \div z$, by conventionally interpreted repeated division by "grouping left", i.e. as $(x \div y) \div z$. Less frequently "grouping right" convention is employed, as in $x^{y^z} = x^{(y^z)}$. However, when dealing with the logical operations of equivalence and implication, the meaning is different. When $A, B, C, \ldots$ are equations, or — more generally — statements, the notation $A \Leftrightarrow B \Leftrightarrow C \ldots$ stands for $(A \Leftrightarrow B) \wedge (B \Leftrightarrow C) \ldots$ which would correspond to an interlocking pattern of parentheses that is too easy to confuse for something else to use in practice:

$$(A \Leftrightarrow [B) \Leftrightarrow (C] \ldots .$$

In a sense, the implication and equivalence signs behave more like an equal sign than an operation symbol.

2.5 History: Stoics

As we undertake our first brief incursion into the subject of history, I want to make a general remark that will pertain to all historical sections in these notes.

Deep ideas are like rivers. When they gather enough strength to get noticed and recognized, their full content is rarely the result of an output from a single source. More typically, they manifest a confluence of many streams of thought, sometimes even contributed at the same time by independent

thinkers, and often running underground completely hidden from an observer, only to reappear later as a crucial admixture in a bigger stream. This makes it difficult to definitively attribute any profound theory or an idea to a single author or a point in time. The best I hope to accomplish in my historical notes is to mark the development of logic not by building monuments at the symbolic but often meaningless origins, but by taking scenic shots in places where the confluence and synergy of already full-bodied ideas created new depth evident enough to be recognized and admired.

While the ideas of propositional logic can be traced at least as far back as Aristotle[12] and Tyrtamus Theophrastus[13], they reached the level of a developed system of reasoning in the works of another Greek philosopher, Chrysippus of Soli.

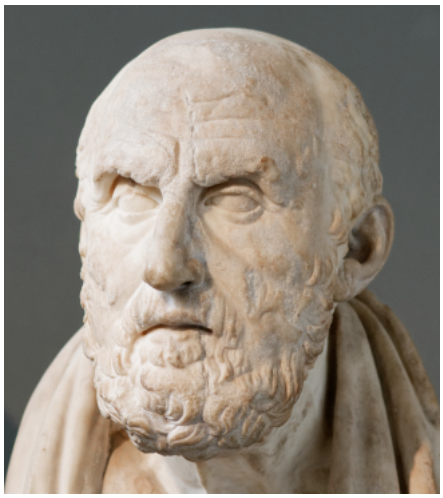


Figure 2: Χρύσιππος [Chrysippos] (c.279–c.204 BC)

Regarded as the leading logician during his own life time, Chrysippus was overshadowed by Aristotle in subsequent history. Chrysippus already had the notions of propositions, logical gates, and argument forms. Roughly speaking, propositional logic is synonymous with "Chrysippus logic".

[12] more about him later

[13] c. 371 – c. 287 BC, Greek philosopher of Peripatetic School who succeeded Aristotle as the school's head.

3 Syllogistic Logic

3.1 Ontology: Categories and Quantifiers

Syllogistic logic shifts the focus away from logical gates (even though it cannot completely dispose of them), and adds the notions of `category`[14] and `quantifier` to propositional analysis of arguments. Consider the following argument:

Example (Syllogistic Argument): All cats are mammals. All mammals are vertebrates. Thus all cats are vertebrates. $\diamondsuit$

Propositional analysis yields:

$$\Big(\text{(all cats are mammals)} \wedge \text{(all mammals are vertebrates)}\Big) \Rightarrow \text{(all cats are vertebrates)}\,.$$

If we replace different statements with different letters, we will get an invalid argument:

$$\Big(C \wedge M\Big) \Rightarrow V$$

HOMEWORK: Why is this argument invalid?

3.2 Apologia: Euler-Venn Diagrams

We have faced a similar situation earlier when a valid propositional argument appeared invalid because we did not go deep enough in its analysis (see page 12). Here, however, deeper *propositional* analysis is impossible: there are no parts in the statements (that we can break away as *full statements*) matching similar parts in other statements.

[14] Note that the word "category" came to mean something entirely different in modern mathematics. The mathematical concept which is the most accurate representation of a category in the sense we use here is the concept of a `set`.

Intuitively, the original argument should be valid — it is the propositional logic that is too crude of a tool to reveal enough of the structure of this argument to show its validity. To capture the connections between individual statements in our analysis, we need to consider building blocks smaller than full statements. Noticing that the words "cats", "mammals" and "vertebrates" are each shared by two of the three statements, we can use those words to pin down the interlocking relation among the statements. Thinking about these words as `categories` of objects in some universe, we can represent objects as points on the plane and those categories — as the domains on that plane. This way we can represent the argument in a graphic form:

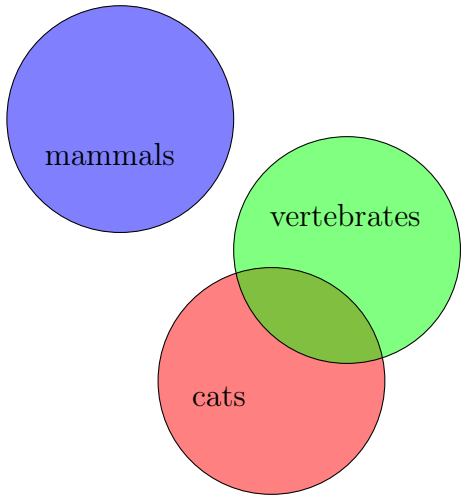


Figure 3: Sketch of the Categories — Too Presumptuous

But wait... This picture seems to imply that no mammal can be either a vertebrate or a cat. When making an initial sketch of the categories involved in an argument, we must avoid the possibility of imparting our picture with any assumptions not postulated in that argument. In other words, we need to draw the disks in *common position*. This idea leads to the following definition:

Definition (of `Venn diagram`): An arrangement of sets on the plane that makes any combination of membership status possible is called a `Venn diagram` of those sets. ♢

HOMEWORK: How many membership possibilities are there for three sets? For n sets, where $n \in \mathbb{N}$?

The following sketch is an example of a Venn diagram for the three categories considered in our argument:

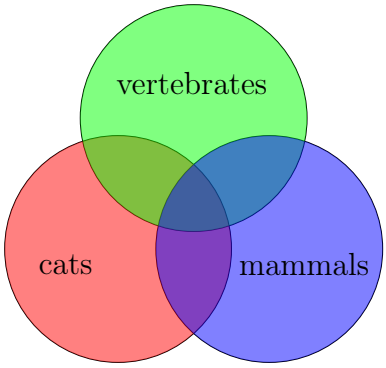


Figure 4: Venn Diagram of the Categories

We can encode the fact "all cats are mammals" by horizontally shading the part of "cats" category lying outside of the "mammals" category:

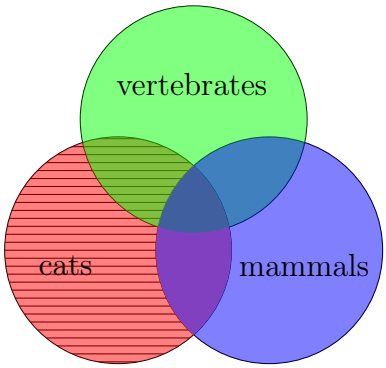


Figure 5: Venn Diagram with One Premise Marked

The shading indicates that no object is permitted to be in the shaded area. Continuing with our analysis, we can encode the fact "all mammals are vertebrates" by vertically shading the part of "mammals" category lying outside of the "vertebrates" category:

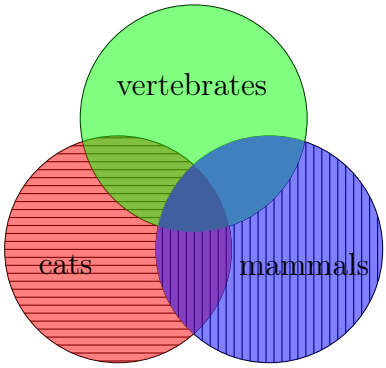


Figure 6: Venn Diagram with Both Premises Marked

This analysis shows that the real picture of categories mentioned in this particular argument looks like this:

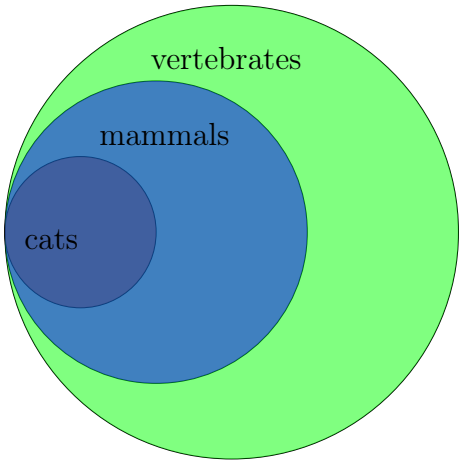


Figure 7: Euler Diagram of the Syllogism

This way of representing the information at hand is called an `Euler diagram`. In contrast with Venn diagrams, which start with an assumption-free depiction of the categories, Euler diagrams summarize the results of Venn diagram analysis by showing the actual configuration of categories reflecting the assumptions of the argument being analyzed.

The above Euler diagram demonstrates the validity of our argument by translating the statements "everybody of this kind is of that kind" into geometric statements "this category is *inside* of that category".

The conclusion that "cats" are inside "vertebrates", is obviously supported by the assumptions that "mammals" are inside "vertebrates" and "cats" are inside "mammals".

Example (syllogistic argument that depends on existential presupposition): Since all unicorns are mammals, and all mammals are animals, we can conclude that some unicorns are animals. $\diamondsuit$

This example corresponds to the following Venn diagram, where the vertical shading represents the assumption "all unicorns are mammals", and the horizontal one — the assumption "all mammals are animals":

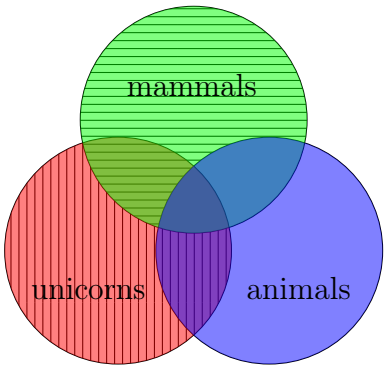


Figure 8: Venn Diagram of the Syllogism

Optionally, the same information can be represented by the Euler diagram

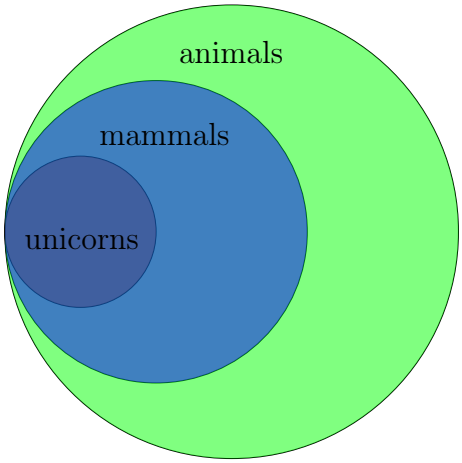


Figure 9: Euler Diagram of the Syllogism

The conclusion we want to test is whether or not this arrangement of the categories guarantees existence of at least one object that would be both a unicorn and an animal.

This example exposes one potential ambiguity in interpreting everyday language. In our particular case, does the phrase "all unicorns are mammals" imply existence of unicorns?

Definition (of `existential presupposition`): The convention stipulating the presumption of existence of an entity mentioned in a noun phrase within a factual[15] context, is called `existential presupposition`. ♢

Existential presupposition — just like any other accepted, but not explicitly stated assumption — may cause many errors in logical reasoning. It is better to avoid it, agreeing to resolve this ambiguity in meaning by the requirement of stating the existence explicitly. Thus, for the rest of these notes, we adopt the convention of *rejecting* the existential presupposition. When we say "all unicorns are mammals", that will mean is really this: "all unicorns — if they exist — are mammals". With our convention in effect, this statement is true when applied to the world we live in — the world with no unicorns — since an implication with a false assumption is always true. Somewhat paradoxically, in our world the statement "all unicorns are not mammals" is also true.

[15] as opposed to counterfactual

With existential presupposition rejected, the above picture may reflect the situation with (some or all) of the categories being empty. Taking empty "unicorns" category (which happens to be the case in the real world) provides a counterexample to this argument. This counterexample shows that the conclusion "some unicorns are animals" is not supported by the assumptions and thus this argument is invalid.

HOMEWORK: We just realized that "some unicorns are animals" is a false conclusion from the premises of this argument. Can we conclude that "all unicorns are animals"?

Example (syllogism): Some people understand logic, but dogs are not people, therefore no dog understands logic. ♢

We start our analysis with a Venn diagram:

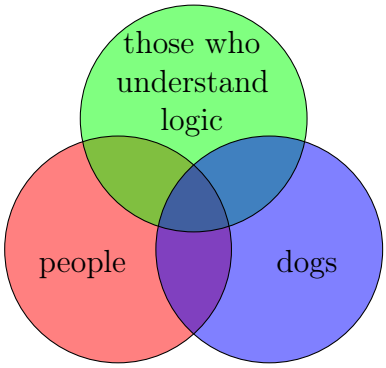


Figure 10: Starting Venn Diagram of the Syllogism

The assumption "some people understand logic" can be visually expressed by drawing an interval within the intersection of "people" and "those who understand logic". The interval is a visual way to indicate the existence of an object (somewhere along that interval) without making an unwarranted assumption about the specific location of that object. The interval, as opposed to a point, indicates that the object in question may be on either side of the "dogs" boundary. [16]

[16]Similarly, in quantum mechanics one gives up on identifying the precise location of a

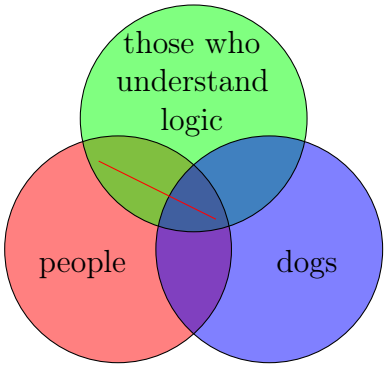


Figure 11: Venn Diagram of the Syllogism with the Existence Clause Marked

The assumption "dogs are not people" can be expressed in the familiar way, namely by shading the region where nothing is permitted to be. In this case, the forbidden region is the intersection of "people" and "dogs": Note that the forbidden region removes the ambiguity expressed by the interval notation, which can now be replaced by a single point.

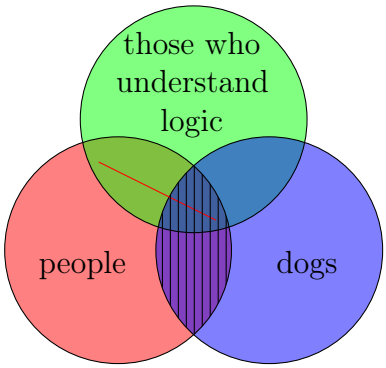


Figure 12: Completed Venn Diagram of the Syllogism

The same information can be expressed by the following Euler diagram:

quantum particle, settling instead for a region where the particle is likely to be found.

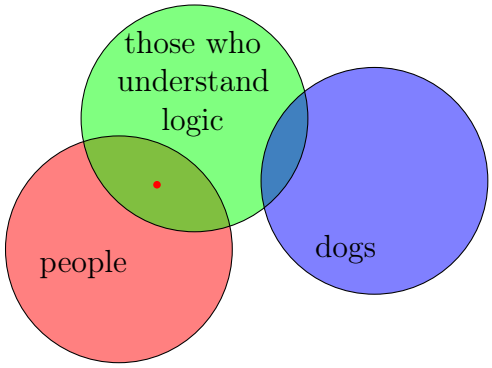


Figure 13: Euler Diagram of the Syllogism

Recall that we want to test the conclusion "no dog understands logic". This conclusion does not follow from the assumptions. Indeed, imagine a world where in addition to the object marked by the red dot (that object must exist because of the assumptions), there is just one more object depicted by the yellow dot in this Euler diagram:

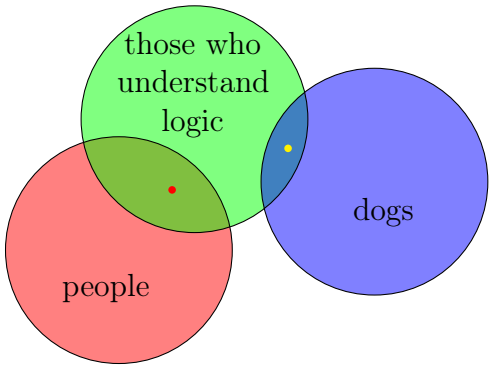


Figure 14: Counterexample for the Syllogism

In other words, make the world with only two objects: one person who understands logic (the red dot above), and one dog who understands logic (the yellow dot above). In that world, the assumptions of this argument are satisfied, but the conclusion is false. Thus this world represents a counterexample for this argument, showing the argument to be invalid. (It is entirely beside the point that such a world does not look like our real world. We are considering the argument in its formal sense, disregarding its connection to reality.)

We managed to get through most of Syllogistic Logic without giving precise definition of what a syllogism was. There is a reason for that. The concept of syllogism is not organic[17], in the sense that it does not grow out of the meaning and structure of logic. Instead, it is a (somewhat artificial) class of problems that can be resolved by the kinds of Venn and Euler diagrams considered in this section. These techniques limit the number of categories to three: while similar Venn diagrams can be constructed for arguments with more categories, the pictures quickly get too complicated to be worth the effort.

3.3 History: Aristotle

Aristotle was a Greek philosopher and the founder of the Peripatetic School. He is credited with establishing logic as a discipline.

Figure 15: Ἀριστοτέλης [Aristotle] (384–322 BC)

Aristotle may have been the first to recognize that the validity of an argument may result from its mere form rather than meaning. He systematically studied syllogisms and discovered some of the ideas of propositional logic

[17]It is not organic in the current sense of the word "organic". It is the epitome of something organic if that word is used in its original sense, as a reference to Aristotle's Organon.

(which historically came after the syllogistic one), like the law of excluded middle and the law of contradiction. Thus, syllogistic logic is also called "Aristotelian logic". Aristotle's treatise of logic, the Organon [2] survived to this day. It provided the foundation for logical studies up to 19th century AD.

The early discovery of this syllogisms by Aristotle and their significance during the antiquity made syllogistic logic one of the cornerstones of liberal arts. The idea of liberal arts [18] can be traced to 4th century BC Greece, where it had at least two distinct roots. The first was the political organization of a Greek polis, or a city-state. Polices had a form of direct democracy that placed great emphasis on the ability of an individual to formulate their ideas and express them in an engaging and convincing way. The second root was the development of mathematics [19] resulting in the idea[20] of mathematical nature of the world. Preserved by the Islamic scholars through the European Dark Ages ensuing from the collapse of the Roman Empire, liberal arts re-emerged in the medieval Europe around the turn of the first millennium, providing the foundation of Western education. By 9th century AD liberal arts were organized into the Trivium (grammar, dialectic and rhetoric) and the Quadrivium (music, arithmetic, geometry and astronomy), with all the seven subjects together comprising philosophy[21].

In the European Renaissance, the disciplines of the Trivium were complemented by history, poetry, ethics and Greek, forming the core of the "Studia humanitatis" or the "humanities" as we know it now. European education from about 1100 to 1700 was based on scholasticism, a philosophy that emphasized joining faith and dialectical[22] reasoning. The idea of dialectics found its later development in Natural Deduction of the Proof Theory, with its characteristic feature of creating the worlds which are permitted to fail in an informative — and thus productive — way. Scholastics used the so-called "critical organic method" of philosophical analysis. That method was based on Aristotle's Organon and placed a big emphasis on the study of syllogisms.

Contrary to the terminology used to describe Euler-Venn diagrams, Gottfried Wilhelm Leibniz used them to analyze syllogisms long before Euler

[18] "artes liberalis", literally "the skills of the free" in Latin

[19] probably also influenced by the Egyptian school of geometry

[20] expressed explicitly by Pythagoras

[21] φιλοσοφία, philosophia, literally "love of wisdom" in Greek.

[22] literally "through conversation" in Greek, meaning "finding the truth through the collision of the opposites"

and Venn.

Figure 16: Gottfried Wilhelm Leibnitz (1646–1716)

In his paper "De Formae Logicae Comprobatione per Linearum ductus", probably written after 1686, Leibniz proposed the creation of a universal language that he called `characteristica universalis` ("universal characteristic" in Latin). That idea inspired Frege to create his Begriffsschrift two hundred years later.

Another major advance in the study of syllogisms came in the works of George Boole. Boole's algebraic treatment of syllogisms in [5] formed the foundation of the algebraization of logic and defined what we now call "Boolean algebra". Boolean algebra studies equations where variables can assume only two possible values: true and false. They are called `boolean variables`.

Figure 17: George Boole (1815–1864)

We continue our study of syllogisms to pay homage to this great tradition, even though the remainder of this section will be superseded by Proof Theory we will introduce subsequently. In addition to their cultural and historical significance, the next few sections may help you see the ideas of the Proof Theory in their historical development, and appreciate those recent advancements more when they emerge against the backdrop of the somewhat murky and tedious workings of the traditional approach you are about to encounter.

3.4 Definition of Syllogism

We could try to define syllogisms by describing their format.

Definition (of `syllogism`): A `syllogism` is a categorical argument that consists of two premises and a conclusion. The premises and the conclusion are called the `clauses` of the syllogism.

The argument must have the following form:

```
<syllogism> ::=
                <clause>
                <clause>
                -----------
                <clause>

<clause> ::= <quantifier> <category> <copola> <category>.

<quantifier> ::= All | Some

<copola> ::= are | are not

<category> ::= ....
```

where the ellipsis in the last production rule stands for the list of the categories specific to syllogism. In our last example, it would be:

```
<category> ::= people | dogs | those who understand logic
```

♢

> **HOMEWORK:** What are the starting, non-terminal and terminal symbols in this syllogism grammar?

However, this definition does not quite do the job: it only specifies the general form of a syllogism while still permitting illegal clauses like "some dogs are dogs".

This is a typical problem in computer science, where programming languages are too complicated to be fully described in terms of their form alone. Defining what constitutes a valid computer program (i.e. an expression) in a given programming language is usually done in several stages. First, using Backus notation, one describes the proper *syntax* of a program. Then, specification of the meaning of constituent commands allows to formulate *semantic* correctness. Finally, the analysis showing that the program does what it is designed to do demonstrates its *pragmatic* correctness.

We could follow that method by adding *semantic* requirements on the top of the *syntactic* definition given on the previous page. Specifically, we can require the categories and the clauses of a syllogism to be related to each other like the vertices and sides of a triangle. Each clause, like a side of a triangle, must contain two different categories, which would be like the vertices of that triangle, with the same incidence structure. There are only four distinct "legal" ways of distributing any three categories among the three clauses of a syllogism, giving rise to the following:

Definition (of `syllogism's figure`): The specific arrangement of any three categories among the clauses of a syllogism is called the `figure` of that syllogism. There are four possible figures, listed in the grammar that appears on the next page. ♢

Since we can list all the figures as separate production rules, syllogisms are, after all, simple enough to be completely described by their syntax alone. Before we do that, two new terms need to be introduced.

Definition (of `subject and predicate of a clause`): In the context of syllogisms, the first category of a clause is called the `subject`, and the second category — the predicate[23] of that clause. ♢

In the following Backus grammar, we shorten "the category that is the subject of the conclusion of the syllogism in question" to "Subject" and "the category that is the predicate of the conclusion of the syllogism in question" to "Predicate". Finally, the third category, which occurs in each premise but is absent from the conclusion, is referred to as the "Middle". At the cost of some redundancy, we also introduce a few additional standard terms:

[23] Later, the word "predicate" will be used in a more general sense.

```
<syllogism>   ::= <FIGURE 1> | <FIGURE 2> | <FIGURE 3> | <FIGURE 4>


<FIGURE 1>    ::= <quantifier> <Middle>    <copula> <Predicate>.
                  <quantifier> <Subject>   <copula> <Middle>.
                  ------------------------------------------------
                  <quantifier> <Subject>   <copula> <Predicate>.


<FIGURE 2>    ::= <quantifier> <Predicate> <copula> <Middle>.
                  <quantifier> <Subject>   <copula> <Middle>.
                  ------------------------------------------------
                  <quantifier> <Subject>   <copula> <Predicate>.


<FIGURE 3>    ::= <quantifier> <Middle>    <copula> <Predicate>.
                  <quantifier> <Middle>    <copula> <Subject>.
                  ------------------------------------------------
                  <quantifier> <Subject>   <copula> <Predicate>.


<FIGURE 4>    ::= <quantifier> <Predicate> <copula> <Middle>.
                  <quantifier> <Middle>    <copula> <Subject>.
                  ------------------------------------------------
                  <quantifier> <Subject>   <copula> <Predicate>.


<quantifier>  ::= <universal> | <existential>
<universal>   ::= All
<existential> ::= Some

<copula>      ::= <affirmative> | <negative>
<affirmative> ::= are
<negative>    ::= are not

<Subject>     ::= ...
<Middle>      ::= ...
<Predicate>   ::= ...
```

The last three production rules need the specific (and distinct) categories in place of the ellipses "...". It is also more traditional to call the existential quantifier the "particular" in the context of syllogisms.

HOMEWORK: Why is the number of possible figures of a syllogism exactly four?

Definition (of `syllogism`): Syllogism is a logical argument, which is constructed according to the grammar specified on the previous page. In determining whether or not an argument is a syllogism, we disregard the issue whether or not that argument is *logically* valid. The only question that needs to be addressed when classifying an argument as a syllogism is its conformance with the above *grammar* rules. ♢

Definition (of `mood of a clause`): The specific combination of a quantifier and a copula is called the `mood` of a clause in a syllogism. There are four possible moods. ♢

HOMEWORK: Why is the number of possible moods of a syllogism clause exactly four?

In traditional medieval scholastics, moods were represented by vowels, with:

- the word "**a**ff**i**rmo" (Latin for "affirm") giving the vowels for the two affirmative moods:
  - **a**: `<universal> <affirmative>`
  - **i**: `<existential> <affirmative>`
- and the word "n**e**g**o**" (Latin for "negate") giving the vowels for the two negative moods:
  - **e**: `<universal> <negative>`
  - **o**: `<existential> <negative>`

Definition (of `major/minor premise`): The first assumption in a syllogism is sometimes called the `major premise`, and the second one — the `minor premise`. ♢

Definition (of `syllogism class`): All syllogisms that are the same, except for possibly their categories, form a `syllogism class`. For example, the syllogism we considered on page 28 belongs to the same class as "all cats are pets, all pets like to play; therefore all cats like to play". Syllogisms that belong to the same class are represented by the same Venn and Euler diagrams. ♢

HOMEWORK: How many syllogism classes are there in total? (Hint: how many moods are there per clause? per syllogism?)

3.4.1 Medieval Mnemonics for Deriving Syllogisms

Syllogisms were given mnemonic[24] names. Each class of syllogisms corresponded to a single word with three vowels (example: "Barbara"). The vowels encoded the mood of each clause, according to the rules on page 44.

For example, the name "Barbara" denoted, by its three vowels "a", the class of syllogisms with all its three clauses having universal affirmative mood. The example introduced on page 28 is an instance of a "Barbara" syllogism.

Here is the full list of the 24 *valid*[25] syllogism classes. Those in italics are only valid with the existential presupposition in effect. Those in non-bold italic admit a stronger conclusion.

[24] i.e. aiding memory retention of certain information

[25] now we are talking about logical validity of an argument

FIGURE 1 **MP, SM**	**FIGURE 2** **PM, SM**	**FIGURE 3** **MP, MS**	**FIGURE 4** **PM, MS**
Barbara	Cesare	Datisi	Calemes
Celarent	Camestres	Disamis	Dimatis
Darii	Festino	Ferison	Fresison
Ferio	Baroco	Bocardo	Calemos
Barbari	*Cesaro*	***Felapton***	***Fesapo***
Celaront	*Camestros*	***Darapti***	***Bamalip***

These names of the syllogisms, together with the exact distribution of categories between the clauses (included above in the table header for each figure), give the full set of valid syllogistic argument classes.

But the name of a syllogism is actually more than just an encoding of its class. It is a mnemonic device, telling how that syllogism can be *justified* using the corresponding syllogism in the first figure.

Figure 1 is called the "perfect figure", so the syllogisms in that figure are also called "perfect". Within this traditional system, imperfect syllogisms are justified by applying the process of `reduction` to them. For each imperfect syllogism, reduction yields the corresponding perfect syllogism, whose validity serves as the last step in the justification process. Reduction is `direct` if the conclusion in the resulting syllogism is equivalent to that in the original, and `indirect` otherwise. It is this reduction process that is encoded in the name of an imperfect syllogism.

The initial letter tells which perfect syllogism corresponds to the imperfect one considered. For example, the initial **D** in "**D**isamis" indicates that "Disamis" reduces to "**D**arii". Letters "r", "t", "l", "n", and non-initial "b" and "d" don't have any mnemonic meaning. Each non-initial letter **S**, **P**, **K**, **M** and **C** defines one transformation step.

Transformations **S**, **P** and **K** apply to the single clause denoted by the previous vowel, while **M** and **C** — to the whole syllogism.

- **S**: "**S**i**m**pl**e**x conversio", or simple clause conversion, is the interchange of the subject and the predicate. It yields an *equivalent* proposition for clauses with **i** or **e** moods. Examples: an **i**-mood clause "some vertebrates are fish" is equivalent to "some fish are vertebrates"; an **e**-mood clause "all humans are not dogs" (usually stated as "no human is a dog") is equivalent to "no dog is a human".

- **P**: "**P**er **a**ccid**e**ns conversio", or partial clause conversion[26], is the interchange of the subject and the predicate, combined with the reversal of the quantifier. With existential presupposition in effect, it yields a *consequence* of the original clause for universal clauses, namely those with moods **a** and **e**. Examples: an **a**-mood clause "all fish can swim" has a consequence "some of those who can swim are fish", as long as we accept that merely mentioning fish in the affirmative clause means that fish exist. Similarly, an **e**-mood clause "all dogs are not birds" (usually stated as "no dog is a bird") has a consequence "some birds are not dogs" when the existential presupposition is in effect.

- **K**: clause obversion, is the complementation of the predicate category, combined with the reversal of the copula. It yields an *equivalent* proposition, for clauses in all four moods. Example: the clause "some vertebrates are fish", has the obverse "some vertebrates are not non-fish".

- **M**: "**M**utatio syllogism", is the interchange the premises. Example: "all mammals are vertebrates; all cats are mammals; thus all cats are vertebrates" when mutated gives "all cats are mammals; all mammals are vertebrates; thus all cats are vertebrates". Since the premises are joined by an implicit conjunction, and conjunction is commutative, the mutated syllogism is *equivalent* to the original one.

- **C**: "**C**onvertio syllogism", the only indirect reduction, is the use of the negation of the original conclusion — as the new minor premise, and the negation of the original minor premise — as the new conclusion. Negation of a clause corresponds to the reversal of both the quantifier

[26] in this context, the word "partial" means limitation of the claim expressed by the original clause

and the copula of that clause. (This rule will be explained in more detail on page 72.) Example: the negation of "some vertebrates are not dogs" is "all vertebrates are dogs".

Indirect conversion is only used for Baroco and Bocardo, and reduces both of these syllogisms to Barbara. Medieval logicians considered it to be as an instance of "reductio ad absurdum", in other words, proof by contradiction. They viewed it as a reduction *through* Barbara, rather then reduction *to* Barbara. (The following homework problem illustrates that conversion can be viewed as an equivalence transformation of the original syllogism as well.)

HOMEWORK: Demonstrate that the convertio syllogism is equivalent to the original one. (SOLVED)

In the original syllogism, denote the major premise P, the minor premise p, and the conclusion C. Converting implications $X \Rightarrow Y$ into $(\neg X) \vee Y$ and using the De Morgan law to distribute the negations of the antecedents (with some trivial adjustments based on associativity and commutativity of disjunction) we get

$$\Big((P \wedge p) \Rightarrow C\Big) \;\Leftrightarrow\; \Big(\neg(P \wedge p) \vee C\Big) \;\Leftrightarrow\; \Big((\neg P) \vee (\neg p) \vee C\Big)$$

for the original syllogism, and

$$\Big(\Big(P \wedge (\neg C)\Big) \Rightarrow (\neg p)\Big) \Leftrightarrow \Big(\neg\Big(P \wedge (\neg C)\Big) \vee (\neg p)\Big) \Leftrightarrow \Big((\neg P) \vee (\neg p) \vee C\Big)$$

for the convertio syllogism. Since the results of both equivalence chains are the same, we can conclude that the original and the convertio syllogisms are equivalent.

Let's consider the traditional justification of the Baroco syllogism:

```
All dogs are mammals.
Some vertebrates are not mammals.
-------------------------------------
Some vertebrates are not dogs.
```

Assume its conclusion is false, meaning that its negation, "all vertebrates are dogs", is true. In that case, the assumptions of the convertio syllogism

```
All dogs are mammals.
All vertebrates are dogs.
-------------------------------
All vertebrates are mammals.
```

are true. But since the convertio syllogism is Barbara whose validity we accept, it yields the conclusion "all vertebrates are mammals". This conclusion contradicts the minor premise of the original Baroco syllogism, "some vertebrates are not mammals". This contradiction demonstrates that our assumption — of Baroco conclusion's falsehood — was false. In other words, the conclusion of Baroco, as well as the whole Baroco argument, must be true.

The transformations listed on page 47, when applied to the whole syllogism, result either in an equivalent or a stronger[27] claim. Thus, if the end result of the reduction is a syllogism whose validity we are willing to accept, then we are forced to accept the validity of the original syllogism (since it is either an equivalent or a weaker claim). This way, every imperfect syllogism in the table is justified by the incremental validity of each single transformation, combined with the validity of the final Figure 1 syllogism. This idea of incremental justification anticipates *proofs* which we will study in later sections.

[27] When replacing an assumption by its consequence, we are, in effect, turning to a stronger claim, which says that the original conclusion follows from a weaker assumption.

Let's finish our discussion of "Disamis" by showing how its justification is encoded in its name. Here is an example of Disamis syllogism::

```
Some fish are vertebrates.
All fish can swim.
-----------------------------------------------
Some creatures that can swim are vertebrates.
```

As mentioned earlier, the first letter **D** in "**D**isamis" indicates that it will be reduced to "**D**arrii". Letter **S** in position 3 of "Di**s**amis" prescribes simplex convertio of the first clause, resulting in an equivalent syllogism

```
Some vertebrates are fish.
All fish can swim.
-----------------------------------------------
Some creatures that can swim are vertebrates.
```

Letter **M** in position 5 of "Disa**m**is" prescribes mutatio, resulting in an equivalent syllogism

```
All fish can swim.
Some vertebrates are fish.
-----------------------------------------------
Some creatures that can swim are vertebrates.
```

Letter **S** at the end of "Disami**s**" prescribes simplex convertio of the conclusion clause, resulting in an equivalent syllogism

```
All fish can swim.
Some vertebrates are fish.
------------------------------
Some vertebrates can swim.
```

which is a Darii syllogism which we accept as valid. Since we accept the last syllogism as valid, and every syllogism we encountered along the way was equivalent to each other one, the starting Disamis is equivalent to Darii, and thus valid as well.

4 Diversion: Lambda Calculus

Elements of `lambda calculus` appeared earlier, but as a complete system, it was invented by Alonzo Church [8]. He envisioned it as a framework for building foundations of mathematics.

Figure 18: Alonzo Church (1903–1995)

Church was building upon the work of Моисей Эльевич Шейнфинкель [Moses Schönfinkel] who invented combinatory logic [37] and Haskell Curry, who developed it [9].

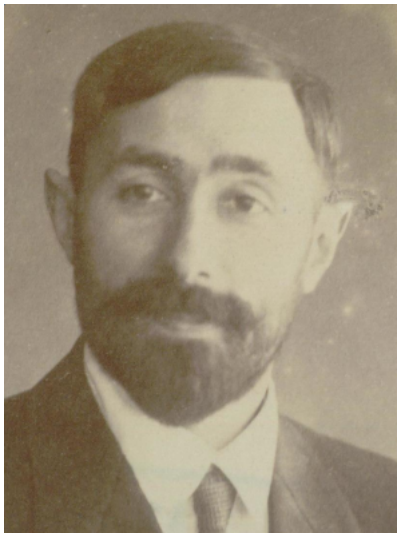


Figure 19: Моисей Эльевич Шейнфинкель [Moses Schönfinkel] (1889–1942)

Figure 20: Haskell Brooks Curry (1900–1982)

However, their original hope came to a crushing defeat: in 1935 Stephen Kleene and J. B. Rosser presented the Kleene–Rosser paradox [26] demonstrating inconsistency of the combinatory logic and lambda calculus[28].

[28] perhaps more precisely, they demonstrated inconsistency of *modeling logic* within combinatory and lambda calculus, rather than inconsistency of those two by themselves.

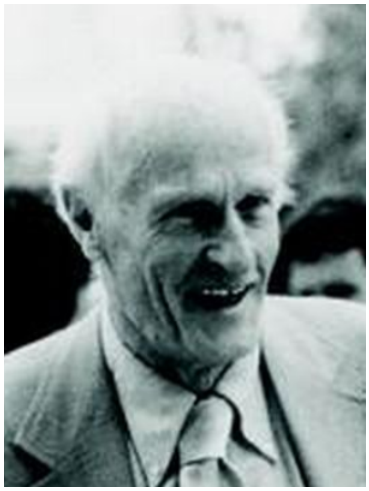


Figure 21: Stephen Cole Kleene (1909–1994)

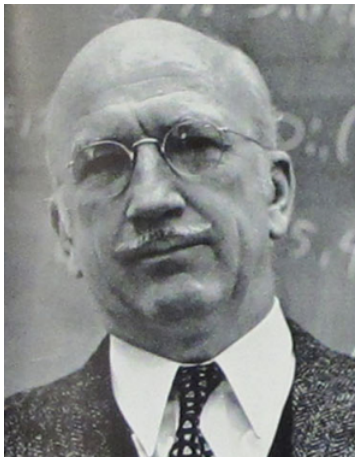


Figure 22: John Barkley Rosser, Sr. (1907–1989)

Kleene–Rosser construction was simplified by in 1942 Curry himself [10] and became known as the Curry's paradox[29].

[29]This footnote, describing the Curry's paradox, is definitely not for the first reading of this text. The original intended use of combinatory logic and untyped lambda calculus was to model predicate logic within these two theories. Those models would give the obvious — predicate — interpretation to the application terms, so that $P(X)$ would mean "X has

Lambda calculus was fixed by Church in 1936 by introducing `types`. That later development of the theory became known as the `simply-typed lambda calculus`.

---

the property P". To model implication, a constant Ξ "ur"-term was added, with $\Xi\left(A(B)\right)$ given the meaning "$A \Rightarrow B$". Within that framework, all logical gates can be represented by lambda terms. (In what follows, we just use the logical gate $\neg$ itself, even though we really mean the lambda term representing it.)

However, both combinatory logic and untyped lambda calculus have a remarkable object, called the Y-combinator, which, for every term F, has the property

$$Y(F) = F\left(Y(F)\right).$$

In a sense, the Y finds the `fixed point` for any function — this is why it is called the "fixed point combinator".

Using the Y-combinator, define the term $C = Y(\neg)$ — we denote it C in honor of Curry. Then the property of the Y-combinator tells us that

$$\neg C = \neg\left(Y(\neg)\right) = Y(\neg) = C,$$

which is a contradiction.

Albeit not in its originally intended role, the early — untyped — version of lambda calculus proved to be extremely useful as a foundation of computer science and a particularly convenient and elegant model of computability. John McCarthy's invention [29] of the computer programming language Lisp, based on lambda calculus, in the late 1950's, gave a physical embodiment to that model. Other programming languages similar in spirit to lambda calculus have been created; they fall into what is called the `functional programming paradigm`.

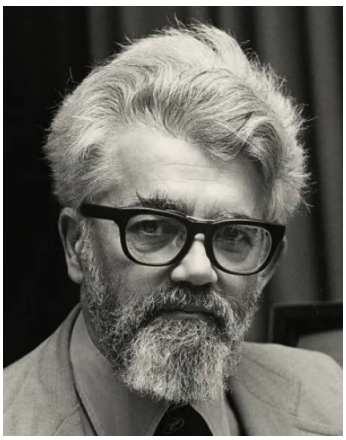


Figure 23: John McCarthy (1927–2011)

4.1 Lambda Notation

The `functional notation` $f(x)$ denotes the value of the function f when that function is given the input x. The functional notation $f(x)$ can be used to define a named function by a formula, as in "define the function $f(x) = x^2$". But often the function is not named, and is simply referred to as "the function x^2". This possibility of calling the expression with x a function creates the basis for taking $f(x)$ as a notation for a function as well. In this example, we are dealing with the square function whose graph is the familiar parabola:

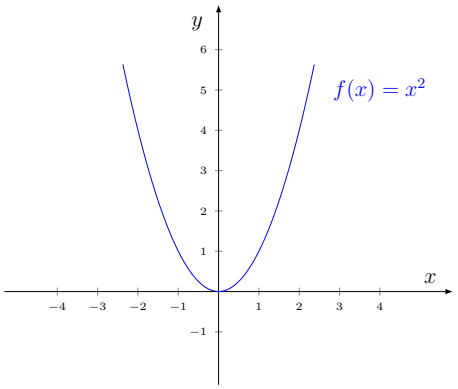


Figure 24: Traditional, but sloppy picture

What happens if $x = 2$? In this case, the $f(x)$ must be 4 — that's the meaning of the functional notation! But then even in the general case (when x is arbitrary) the $f(x)$ should denote a number, not a function. We seem to have cornered ourselves into a contradiction. In spite of the usual custom, we can't really say that $f(x)$ is a function. The $f(x)$ can only denote the number, which is the output of the function f. What is more, it means that our usual way to refer to a function using its formula, as in "the function x^2", is also somewhat illegitimate. How can we rescue this useful but sloppy way of talking about functions? This is exactly the problem that the lambda notation solves.

While the form of lambda notation varies, the meaning of it is the explicit expression of the fact that the whole function rather than its output value corresponding to some input value is being referred to. This concept, regardless of how it is denoted, is called the `lambda abstraction`. In these notes, we will denote the lambda abstraction as

$$\Big(x : f(x)\Big) .$$

Whenever possible to do so without confusion, we will omit the outer parentheses.

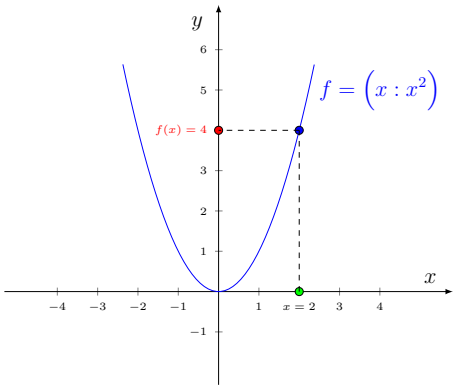


Figure 25: Corrected picture

The original notation used by Alonso Church for the same thing was

$$\left(\lambda x.fx\right),$$

explaining the use of the word "lambda", which is the name of the Greek letter λ. Modern computer programming languages usually express the same concept with something more verbose:

```
function ( x ) {
    return x * x;
}
```

with some allowing a shorter form `x => x*x` as well. One advantage of the more verbose lambda notation (and perhaps the main reason it is often preferred in programming) is the possibility of using it for giving a name to the function being defined.[30] The mathematical idea of taking the function $f(x) = x^2$ would be expressed in programming as

```
function f( x ) {
    return x * x;
}
```

while the idea of taking the (anonymous) function x^2" corresponds to the already mentioned

[30]Lambda calculus does have the expressive power to represent the idea of naming *constant* objects. We will discuss it in more details in the section devoted to contexts and models, which starts on page **??**.

```
function ( x ) {
    return x * x;
}
```

The named function can be applied, whenever we need it, using the notation `f( 5 )`. We could use any other name in place of "`f`". For example, if we define

```
function square( x ) {
    return x * x;
}
```

then the "`square`" and "`f`" will be the same. We will use this format for naming in our notation for proofs.

With lambda notation, we can make these things precise: $f(x)$ by itself will be used to denote the value of the function f corresponding to the input x, while

$$\Big(x : f(x)\Big)$$

will describe the (whole) function that takes the input x into the output $f(x)$. Thus $(x : x^2)$ is the parabola, and x^2 is an individual number on the y-axis, which is the square of some other fixed number x on the x-axis.

4.2 Lambda Calculus Grammar

The untyped lambda calculus deals with expressions constructed according to the following grammar rules:

```
<term> ::= <variable> | <abstraction> | <application>

<abstraction> ::= ( <variable> : <term> )

<application> ::= <abstraction> ( <term> )

<variable> ::= ...
```

I hope you recognize that the abstraction is exactly the lambda notation, and the application is the functional notation.

Example (well-formed lambda term):

$$\Big(x : x(y)\Big)\,\Big((z : z)\Big)$$

The first set of big parentheses defines a function $\Big(x : x(y)\Big)$ which takes a function and applies it to some fixed y. Note that this is a function on functions! The second set of big parentheses contains the identity function $(z : z)$ which spits back its input as its output. So, the whole thing says: evaluate with the input y the identity function. Thus this expression must equal to y. $\diamondsuit$

In pure lambda calculus, all objects are terms, in other words lambda expressions themselves. However, in "the real world", lambda calculus is usually used not in its pure form, but as an addition to some other underlying reality[31]. In these notes, we allow ourselves to use variables, constants and externally defined functions which are not lambda terms. This will permits us to talk about things like the square function $\Big(x : x^2\Big)$ and numbers, as in

$$\Big(x : x^2\Big)\,(5).$$

This last expression says "apply the square function to 5", so it must equal 25.

4.3 Lambda Calculus Reductions

To clarify how one lambda expression can be turned into another, we can define some purely syntactic reduction rules. The remarkable fact about those reduction rules [32] is their ability to encompass what the word "computation" means: from the point of view of lambda calculus, it means reducing an expression to an equivalent expression of simpler form.

1. β-`reduction`. Example:

$$\Big(x : x^2\Big)\,(y) = y^2.$$

[31] This is very similar to the distinction between axiomatic set theory, where every object is a set, and the "naive" set theory, which is permitted to consider some external objects, called `urelements`, as in $\{1, 2, 5\}$.

[32] The precise and definition requires a bit more care. Instead of diving into the technical details, we illustrate the rules with examples.

In other words, if our function takes x into x^2, then when applied to y it will take it to y^2. This is the precise meaning of the idea of substitution.

2. α-`conversion` (sometimes called α-`equivalence`). Example:

$$\Big(x : x^2\Big) = \Big(y : y^2\Big).$$

It says that a function is defined by its action on its input, not by how its input is denoted. This allows us to choose arbitrary (and preferably meaningful) names for our variables.

3. η-`reduction`. Example:

$$\Big(x : f(x)\Big) = f.$$

This is merely saying that the abstraction defining the function that takes x into $f(x)$ does not define anything new: it is the same thing as the function f itself.

HOMEWORK: Compute

$$\Big(y : (x : \sqrt{x})(y)\Big)\,(49).$$

5 Predicate Logic

5.1 Ontology: Objects and Predicates

Syllogistic logic was one of the cornerstones of traditional European education for millennia, and served as a foundation of liberal arts. In spite of its place in our culture, I suspect (and even hope) that our earlier foray into this subject left you yearning for a better way to think about thinking. Indeed, syllogistics cannot be a complete system of rational reasoning: its rigidity limits its expressive ability, its rules of inference rely on our ability to draw pictures, and it seems somewhat disconnected from propositional logic. We need a system that would, at a minimum, unify propositional and syllogistic logic and provide a common way to validate arguments of both types.

A grand unification of these two subjects is possible and is given by `predicate logic`, sometimes also called `first order logic`. Predicate logic was the creation of the German mathematician Gottlob Frege, who in 1879 in [13] introduced the "Begriffsschrift", (or a "formal language" in German) for expressing mathematics[33].

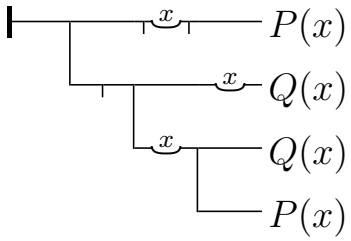


Figure 26: A Predicate Logic Argument Written in Frege's Begriffsschrift

(More details about the above argument can be found on page 94.)

[33]For the second reading of this text: Frege did not separate objects and predicates in his original description of the Begriffsschrift, thus his language allowed quantification over predicates as well as objects. This fact makes some people inclined to argue that Frege introduced *second* order logic. However, given that it was a bug and not a feature of his approach, and the theory got eventually corrected by separation of objects and quantifiers, we view Frege's Begriffsschrift as a first draft of the *first* order logic.

Frege's work laid the foundation for logistical method and the deductive systems we use today.

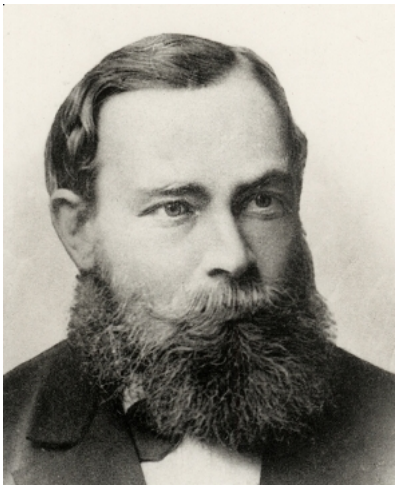


Figure 27: Friedrich Ludwig Gottlob Frege (1848–1925)

In predicate logic, the concepts of

- `objects`;
- `predicates`; and
- `quantifiers`

are added to the familiar statements and gates of propositional logic. (Additional entities, like functions, can be added later as we develop this subject.)

Definition (of `object`): An `object` is a formal model for any *thing* or *entity* we talk about. For example, in our previous syllogism example, we can consider individual people, dogs and creatures who understand logic as objects. $\diamondsuit$

Definition (of `predicate`): A `predicate` is a property of an object or several objects. In our previous syllogism example, "being human" and "understanding logic" are predicates. Similarly, being a dog simply means being an object with the property of "being a dog". We can see that predicates can express the idea of a category. However, a predicate is more general than

a category, because a predicate may be a property pertaining to several objects, and thus expressing a relation among them. For example, "is a parent of" is a predicate characterizing some property of a pair of objects, namely the fact of their familial relation. ♢

The functional notation, as well as the lambda abstraction, can be used with predicates in the same way they are used with numerical functions. This use can be explained by viewing a predicate as a particular type of a function, namely the one taking (zero or more) objects as inputs and returning the truth value "true" or "false" as the output. In fact, the functional notation is often preferred in logic, so that when dealing with a predicate "is a dog",

is a dog(Sparky)

would be used instead of the more natural "Sparky is a dog".

Definition (of `monadic predicate`): Properties that apply to a single object are called `monadic`. Monadic predicates are the predicate logic translation for the categories of syllogistic logic: a category can be now seen as the set of all objects having the property expressed by the corresponding monadic predicate. For instance, the category of dogs is the set of all objects having the property of "is a dog". The property "is a dog" is the monadic predicate corresponding to the category of dogs. [34] Given a monadic predicate P, the set of all objects that have property P will be denoted $[P]$. [35] ♢

5.1.1 Russell's Paradox

Originally Frege did not impose any separation between the objects and the predicates of his logical theories. In that setup, it was quite possible to apply a predicate to another predicate. Bertrand Russell noticed, in his 1902 letter to Frege [36], that this unrestrained freedom allowed the self-reference needed for recreating the liar's paradox in the context of predicate logic.

[34] Frege called monadic predicates "the concepts" and their corresponding categories the `extensions` of the concepts. That's why in the set theory the axiom postulating the existence of the set $[P]$ is called the "extensionality" axiom.

[35] This deviates from the usual `set builder` notation. When the $x : P(x)$ is interpreted as the lambda abstraction (as we do in these notes), the η-reduction makes $\Big(x : P(x)\Big) = P$. This, in turn, gives $\Big\{x : P(x)\Big\} = \{P\}$, breaking the meaning of the set builder notation which — by this logic — must now denote the set containing the predicate P, rather than the objects that have the property P, as usually intended.

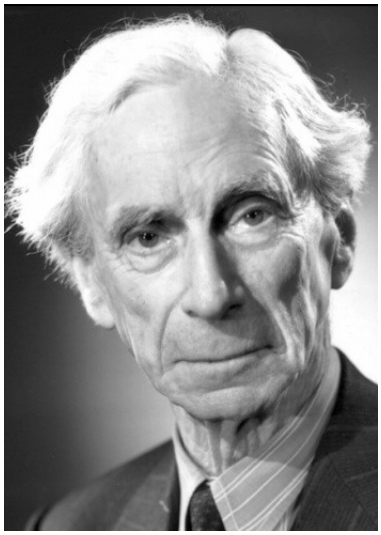


Figure 28: Bertrand Arthur William Russell (1872–1970)

Russell's original argument, now known as "the Russell's paradox", goes as follows. Suppose that predicates can be applied to other predicates. Consider the predicate (which we named R in honor of Russell):

$$R = \Big(P : \neg P(P)\Big).$$

The predicate R, when applied to a predicate P, states that P characterizes a property the P itself does not have. For instance, the predicate

$$A = (\text{can be defined in less than 10 words})$$

itself can be defined in less than 10 words, thus $A(A)$ is true, meaning that $R(A)$ is false. On the other hand, the predicate

$$B = (\text{is a dog})$$

is not itself a dog, thus $B(B)$ is false, and thus $R(B)$ is true. Now Russell asks whether the statement $R(R)$ is true or false. By definition of R, for any predicate P, we have that $R(P) = \neg P(P)$. Taking $P = R$, we get:

$$R(R) = \neg R(R),$$

which is a contradiction.

In a sense, this is exactly the same phenomenon as the liar's paradox, as well as the Curry's paradox of combinatorial/lambda calculus (as well as few other cases mentioned in the history timeline section of these notes). Similar to the fix of lambda calculus, the Begriffsschrift got rescued by separating objects from predicates, and by restricting quantifiers to object quantification only. This fix, together with a change in the notation, turned the Begriffsschrift into the modern predicate logic.

Monadic predicates also allow us to define complements of categories.

Definition (of `complement of a category`): A category defined by the negation of the predicate P:

$$x : \Big(\neg P(x)\Big)$$

is called the `complement` of the category defined by the P itself. $\diamondsuit$

This idea of a monadic predicate can be generalized to any $n \in \mathbb{N}$:

Definition (of n-`adic predicate`): For any $n \in \mathbb{N}$, an n-adic predicate is a property that applies to n objects. (For $n = 2$, a 2-adic predicate is often called `dyadic`.) $\diamondsuit$

Example (n-adic predicate for $n = 0$): We have encountered 0-adic predicates before: those are just statements. So, in addition to categories predicate logic can express the idea of a statement as well: statements are simply the null-adic predicates. $\diamondsuit$

Since we incorporate all logical gates into predicate logic, we can already express within it everything we wish to do in propositional logic. To complete our ability to do the same for syllogistic logic, we need to:

1. generalize logical gates to predicates (rather than just statements) and
2. introduce quantifiers.

5.1.2 Applying Logical Gates to Predicates

What does it mean to use gates with predicates? For example, starting with two predicates "is a dog" and "understands logic", we can form another predicate "is a dog who understands logic", which corresponds, in a more pedantic way, to the conjunction of the original two:

$$\text{is a dog} \wedge \text{understands logic.}$$

But this expression is somewhat ambiguous. Are we talking about one object being both a dog and understanding logic, or two objects, the first being a dog and the second one understanding logic? This problem is compounded by the possibility of the predicates applying to varying number of objects. In case of monadic predicates, we can easily agree that it should mean the monadic predicate

$$\Big(x : \underline{\big((x \text{ is a dog}) \wedge (x \text{ understands logic})\big)}\Big).$$

But the lambda notation opens a different, much more ambitious possibility. In effect, the lambda abstraction creates a context within the underlined part of the lambda expression where x is a *constant* object, and thus "x is a dog" and "x understands logic" are just *statements*. So, we can apply conjunction to those statements just like we did earlier in propositional logic. This idea of context allows the use of several constant objects (e.g. x, y, etc.) in place of one x, thus extending the application of logical gates from predicates to `formulas`. A formula is a building block of a predicate, a "predicate in waiting", namely an expression which may or may not have variables in it and could serve as the right hand side of a lambda abstraction. Consider, for example, this definition:

$$\Big((x, y) : \big((x \text{ is a dog}) \wedge (y \text{ understands logic})\big)\Big).$$

It describes a dyadic predicate that applies to two objects and says that the first one is a dog and the second understands logic.

5.2 Ontology: Quantifiers

Definition (of `quantifier`): In the narrow and literal sense, a `quantifier` tells how many objects in our universe of discourse have the property ex-

pressed by that predicate. Generalizing, we can view a quantifier as any property of a predicate. This more general idea is certainly not a part of the predicate logic in the classical sense, but could be helpful as a way organizing things into a *hierarchy* of abstractions: objects, properties of objects (i.e. predicates), properties of properties of objects (i.e. quantifiers), ... This hierarchy also shows the direction where predicate logic can grow to gain an even stronger expressive power. Applying a quantifier to a predicate results in a statement, just like applying a predicate to an object does. $\diamondsuit$

Classically, predicate logic deals with just the following two quantifiers.

Definition (of `universal quantifier`): The `universal quantifier` states that the predicate applies to all objects in our universe of discourse. It is denoted by the symbol $\forall$. For example,

$$\forall x : x \text{ understands logic}$$

states that everybody understands logic. Note that we wrote "$x : x$ understands logic" instead of the simpler "$\forall$ understands logic". The more verbose notation is traditionally favored when using quantifiers. $\diamondsuit$

Definition (of `existential quantifier`): The `existential quantifier` states that the predicate applies to at least one choice of an objects (or objects) in our universe of discourse. It is denoted by the symbol $\exists$. For example,

$$\exists x : x \text{ understands logic}$$

states that somebody — at least one person — understands logic. Like before, the original predicate "understands logic" is monadic, and the resulting one is a statement. $\diamondsuit$

5.2.1 Bounded Quantifiers: Syllogisms within Predicate Logic

To translate syllogisms into the language of predicate logic, we need to adjust the general purpose existential and universal quantifiers a bit. Specifically, we want to express the idea of those quantifiers quantifying not over all possible objects in our universe of discourse, but rather over the objects of some category. These versions of the universal and existential quantifiers are called `bounded`.

For example, instead of the general purpose universal quantifier saying "all x have the property P":

$$\forall x : P(x),$$

we want to say "all x in *certain category* have the property P". It turns out that we don't really need anything new: we just need to combine the unbounded universal quantifier with the some logical gates. Denote C the predicate whose extent is the category $[C]$ to which we want to bound our quantifier. Then "all x in the category $[C]$ have the property P" can be expressed as

$$\forall x : \Big(C(x) \Rightarrow P(x) \Big)$$

The implication is exactly the right tool to express the fact that we don't care about whether or not the x has the property P if that x fails to be in the category C. The bounded universal quantifier is often written using set-theoretic notation as

$$\forall x \in [C] : \Big(P(x) \Big) .$$

Analogously, to express that "some x in $[C]$ have the property P", we can use

$$\exists x : \Big(C(x) \wedge P(x) \Big) .$$

The symmetry of the two predicates C and P in this expression explains the earlier equivalence of the original clause and its simplex conversio, discussed on page 47: simplex conversio just switches the P and the C, which has no effect on the truth value of the whole statement because of the commutativity of adjunction. The bounded existential quantifier can also be written in set-theoretic notation as

$$\exists x \in [C] : \Big(P(x) \Big) .$$

Example (syllogism translated into predicate logic): Consider our first example of a syllogistic argument: "All cats are mammals. All mammals are vertebrates. Thus all cats are vertebrates."

Let's try to translate it into the language of predicate logic. The categories are now represented by monadic predicates, and the quantifiers of syllogistics become the bounded quantifiers of the predicate logic. The whole argument becomes

$$\begin{array}{l} \forall x : x \text{ is a cat } \Rightarrow x \text{ is a mammal} \\ \forall x : x \text{ is a mammal } \Rightarrow x \text{ is a vertebrate} \\ \hline \forall x : x \text{ is a cat } \Rightarrow x \text{ is a vertebrate} \end{array}$$

Perhaps a less intuitive, but a bit more technically precise way to write the same thing would be to use the functional notation:

$$\begin{array}{l} \forall x : \text{cat}(x) \Rightarrow \text{mammal}(x) \\ \forall x : \text{mammal}(x) \Rightarrow \text{vertebrate}(x) \\ \hline \forall x : \text{cat}(x) \Rightarrow \text{vertebrate}(x) \end{array}$$

This is one step away from the "model free" view of this argument. Analogous to replacing *statements* with generic variables in a propositional argument, we can replace with generic variables the *predicate names* referring to the incidental details of our specific model. Such replacement makes the underlying logical structure of our argument more transparent:

$$\begin{array}{l} \forall x : C(x) \Rightarrow M(x) \\ \forall x : M(x) \Rightarrow V(x) \\ \hline \forall x : C(x) \Rightarrow V(x) \end{array}$$

$\diamondsuit$

Example (syllogism): Some people understand logic, but dogs are not people, therefore no dog understands logic.

Consider the universe of discourse whose objects can be dogs, people and any other creatures (or even things) that understand logic. Each category can be represented in predicate logic as a monadic predicate. So, we have three monadic predicates: "is a dog", "is a human", and "understands logic".

"Some people understand logic" can be expressed as

$$\exists x : x \text{ is a human} \wedge x \text{ understands logic.}$$

"Dogs are not people" means

$$\forall x : x \text{ is a dog} \Rightarrow \Big(\neg(x \text{ is a human})\Big).$$

Finally, "no dog understands logic" can be reworded "every dog does not understand logic" and expressed as

$$\forall x : x \text{ is a dog} \Rightarrow \Big(\neg(x \text{ understands logic})\Big).$$

$\diamondsuit$

5.2.2 Negation of Quantified Statements

Negation of a universally quantified statement results in an existentially qualified one:

$$\neg\Big(\forall x : \Big(P(x)\Big)\Big) \Leftrightarrow \Big(\exists x : \Big(\neg P(x)\Big)\Big)$$

One may view it as a generalization of the De Morgan law. Indeed, imagine the world where there are only finitely many objects, so that we can list them all $c_1, c_2, \ldots c_n$. Then saying that the predicate P holds for every object is the same as the conjunction:

$$\Big(\forall x : \Big(P(x)\Big)\Big) \Leftrightarrow \Big(P(c_1) \wedge P(c_2) \wedge \ldots \wedge P(c_n)\Big).$$

Now that we have a conjunction, the corresponding De Morgan Law tells us how to distribute negation along its terms:

$$\neg\Big(\forall x : \Big(P(x)\Big)\Big) \Leftrightarrow$$
$$\neg\Big(P(c_1) \wedge P(c_2) \wedge \ldots \wedge P(c_n)\Big) \Leftrightarrow$$
$$\Big(\Big(\neg P(c_1)\Big) \vee \Big(\neg P(c_2)\Big) \vee \ldots \vee \Big(\neg P(c_n)\Big)\Big).$$

But the last statement says exactly that the property P does not hold for at least one of the objects $c_1, \ldots, c_n$ — which can be expressed by the existentially quantified statement

$$\Big(\exists x : \Big(\neg P(x)\Big)\Big).$$

To stress that universal quantification may be seen as (possibly infinite) conjunction, Polish school logicians use the notation $\bigwedge_x P(x)$ in place of $\forall x : \Big(P(x)\Big)$.

Similarly, to negate an existentially quantified statement, we need to switch to universal quantification:

$$\neg\Big(\exists x : \Big(P(x)\Big)\Big) \Leftrightarrow \Big(\forall x : \Big(\neg P(x)\Big)\Big).$$

In the Polish school, the notation $\bigvee_x P(x)$ is used in place of $\exists x : \Big(P(x)\Big)$ to underscore the connection of existential quantifier with (possibly infinite) disjunction.

Let's see how these rules specialize for bounded quantifiers. Take the universal quantifier bounded to category C, which we will describe by using a predicate with the same name.

$$\neg\Big(\forall x : \Big(C(x) \Rightarrow P(x)\Big)\Big) \Leftrightarrow \Big(\exists x : \neg\Big(C(x) \Rightarrow P(x)\Big)\Big) \Leftrightarrow \Big(\exists x : \Big(C(x) \wedge \Big(\neg P(x)\Big)\Big)\Big).$$

(We used that falsehood of an implication is the same as simultaneous truth of its assumption and falsehood of its conclusion.) Thus the negation of a predicate, quantified by a bounded universal quantifier is the negation of the original predicate quantified by an existential quantifier bounded in the same way: negation of "every dog understands logic" is "there is a dog who does not understand logic".

Similarly, the negation of a predicate, quantified by a bounded existential quantifier, is the negation of the original predicate, quantified by a universal quantifier, bounded in the same way:

$$\neg\Big(\exists x : \Big(C(x) \wedge P(x)\Big)\Big) \Leftrightarrow \Big(\forall x : \neg\Big(C(x) \wedge P(x)\Big)\Big) \Leftrightarrow \Big(\forall x : \Big(\Big(\neg C(x)\Big) \vee \Big(\neg P(x)\Big)\Big)\Big) \Leftrightarrow \Big(\forall x : \Big(C(x) \Rightarrow \Big(\neg P(x)\Big)\Big)\Big).$$

(We used the De Morgan law to distribute negation over the conjunction.) In other words, the negation of "some dogs understand logic" is "every dog does not understand logic".

One can also write the negation of bounded quantifiers in the alternative, set-theoretic form:

$$\neg\left(\forall x \in [C] : \Big(P(x)\Big)\right) \Leftrightarrow \left(\exists x \in [C] : \Big(\neg P(x)\Big)\right),$$

$$\neg\left(\exists x \in [C] : \Big(P(x)\Big)\right) \Leftrightarrow \left(\forall x \in [C] : \Big(\neg P(x)\Big)\right).$$

The content of this section explains in full detail the rule we mentioned in passing on page 47 for negating a clause in a syllogism: "negation of a clause corresponds to the reversal its quantifier and copula." The reversal of the quantifier is the main point discussed above, the reversal of the copula corresponds to the negation of the original predicate, preservation of the subject corresponds to retaining of the original bounding of the quantifier.

5.2.3 Grammar and Language of Predicate Logic

<formula> ::= <application> | <gate>
| <quantification>

<application> ::= <n adic predicate>(<n variable list>)

<gate> ::= <conjunction> | <disjunction> | <negation>
| <equivalence> | <implication>

<conjunction> ::= <formula> $\wedge$ <formula>

<disjunction> ::= <formula> $\vee$ <formula>

<equivalence> ::= <formula> $\Leftrightarrow$ <formula>

<implication> ::= <formula> $\Rightarrow$ <formula>

<negation> ::= $\neg$ <formula>

<quantification> ::= <existentiation> | <universalization>

<existentiation> ::= $\exists$ <n variable list> : <formula>

<universalization> ::= $\forall$ <n variable list> : <formula>

<0 variable list> ::= (empty expression)

<1 variable list> ::= <variable>

<(n+1) variable list> ::= <variable>, <n variable list>

<variable> ::= x | y | z | ...

<0 adic predicate> ::= P0 | Q0 | R0 | ...

<1 adic predicate> ::= P1 | Q1 | R1 | ...

....

The grammar described on the previous page specifies what it means to be a *formula* of the classical predicate logic. To define proper *expressions* in the language of predicate logic, we need to supplement these syntactic rules with the semantics of what constitutes a *statement*.

Definition (of `free and bound variables`): For a formula of predicate logic, `free` variables are defined inductively in terms of the parse tree of that formula. When the formula in question is:

- an `<application>` having the form

  `<n adic predicate>( <n variable list> )`,

  its free variables are given by the `<n variable list>`;

- a `<gate>` having the form

  `<formula> <operation> <formula>`,

  its free variables are given by the union of the free variables in the two constituent formulas;

- a `<negation>` having the form

  ¬ `<formula>`,

  its free variables are those of the negated formula;

- a `<quantification>` having the form

  `<quantifier> <n variable list> : <formula>`,

  its free variables are those of the quantified formula, less the variables in the `<n variable list>`,

Every variable which is not free (in other words, the one in the scope of one of the quantifiers) is called `bound`. ♢

Definition (of `statement in terms of predicate logic grammar`): In the language of predicate logic, a `statement` is a formula without free variables. Sometimes such a formula is also called a `closed formula`. ♢

Definition (of `the language of predicate logic`): The language of predicate logic is the set of all closed formulas among all of the formulas generated according to the grammar specified on the previous page. ♢

5.2.4 Other Quantifiers

While the classical predicate logic uses only the universal and existential quantifiers, few other examples deserve to be mentioned.

Definition (of `exists unique quantifier`): The `exists unique` quantifier, denoted "$\exists\,!$", is the assertion of existence of a unique object that has the property being quantified. $\diamondsuit$

The interesting aspect of this quantifier is the impossibility of expressing it in terms of what we have. We need the concept of *equality* of objects. With it, the "exists unique" quantifier can be expressed in terms of the two usual quantifiers:

$$\Big(\exists\,!\;x : P(x)\Big) \Leftrightarrow \Bigg(\exists x : \Big(P(x) \wedge \forall y : \Big(P(y) \Rightarrow (y = x)\Big)\Big)\Bigg).$$

It says that there exists *unique* x satisfying property P if and only if an x (unique or not) satisfying P exists, and any other object y with the same property P must equal the original x.

Definition (of `definite description`): When a property P admits the "exists unique" quantifier, we are permitted to select the specific object that is uniquely identified by the property in question. Such selection is called a `definite description` of that object. Bertrand Russell introduced the concept in 1905, and used the notation

$$\iota x : P(x)$$

for the only object that has the property P. There are many interesting aspects of this concept, still debated by philosophers to this day, dealing with the meaning of this construct in a general situation when the "exists unique" quantifier is not asserted for the property P. However, we will restrict our use of the ι selector to those situations where it is guaranteed to work without complications by the established truth of $\exists\,! x : P(x)$. $\diamondsuit$

Example (definition of a function): Recall that a `binary relation` R is a triple of sets: $R = (A, B, C)$ such that $C \subseteq A \times B$. Intuitively speaking, a binary relation is a record of bonds connecting two entities in a way allowing

their differentiation (the last part meaning that we want to be able to distinguish parent John and child James on one side, from parent James and child John on the other). The set A (called the `domain` and denoted $\mathrm{Dom}\, R$) is the list of all the entities originating the bond, the set B (called the `range` and denoted $\mathrm{Rg}\, R$) records all those entities which are receiving the bond, and the set C (called the `graph`[36] and denoted $\mathrm{Gr}\, R$) is the list of all the pairs connected by these bonds. For example, the triple of sets (A, B, C) defined as:

$$\begin{aligned} A &= \{\mathrm{John}, \mathrm{Jane}, \mathrm{James}\}, \\ B &= \{\mathrm{James}, \mathrm{Linda}, \mathrm{Mary}\}, \\ C &= \{(\mathrm{John}, \mathrm{James}), (\mathrm{Jane}, \mathrm{James}), (\mathrm{James}, \mathrm{Linda})\}, \end{aligned}$$

is a binary relation. This binary relation may be interpreted as a record of parent-child bonds in one particular group of people, with the A listing all parents of that group, and B including all children of those in A.

Now, a binary relation f is a `function` if and only if it satisfies the `vertical line test`:

$$\forall x \in \mathrm{Dom}\, f : \Big(\exists\, !\, y \in \mathrm{Rg}\, f : \Big((x, y) \in \mathrm{Gr}\, f \Big) \Big),$$

which is effectively saying that for every "input" x, there exists unique "output" y.

Assuming that f is a function, we can use the $\imath$ selector to define the usual `functional notation` $f(x)$:

$$\forall x \in \mathrm{Dom}\, R : f(x) = \Big(\imath y \in \mathrm{Rg}\, R : \Big((x, y) \in \mathrm{Gr}\, R \Big) \Big),$$

which says "take as the $f(x)$ that unique y which ...". The use of the $\imath$ selector is justified by the existence and uniqueness — postulated in the vertical line test — of the y for every given x. $\diamondsuit$

"Exists unique" quantifier can be generalized to "there are at least n", "there are at most n", "there are exactly n", for any $n \in \mathbb{N}$.

Similar selector can be introduced for the regular existential quantifier. **Definition** (of `Hilbert's epsilon selector`): Suppose that predicate P

[36] in this context, the word "graph" has nothing to do with the words "picture" or "sketch" — it is just a reference to a set which is one particular part of a binary relation

satisfies the existential quantifier:

$$\exists x : P(x).$$

Then the notation ϵP can be used for any one of those x that have the property P. The ϵ is called the `Hilbert's epsilon selector`.

The original Hilbert's definition allowed ϵP to be used (as a reference to an object) even when the P did not satisfy the existential quantifier. Broadening the meaning of the ϵ selector this way, Hilbert expressed the existential and the universal quantifiers in terms of it:

$$\Big(\exists x : P(x)\Big) \Leftrightarrow P(\epsilon P).$$

$$\Big(\forall x : P(x)\Big) \Leftrightarrow P\left(\epsilon x : \Big(\neg P(x)\Big)\right).$$

$\diamondsuit$

Quantifying several predicates cannot be, in general, reduced to quantifying a single one. For example, how can we translate into logic the statement "most cats like to play"? If $C(x)$ says that x is a cat, and $P(x)$ means that x likes to play, what are we trying to say here is that the intersection of the categories P and C contains more than half of the objects of C. Of course this idea can also be generalized, yielding constructions like "more than a third of cats like to play" and the like.

HOMEWORK: Can you see why these concepts cannot be expressed in terms of quantifying a single predicate?

Example (valid argument outside of predicate logic): Consider the argument:

```
Most cats like to play.
Most cats are domestic.
-----------------------------------
Some cats are domestic and playful.
```

Since more than half of all cats like to play (first premise), and more than half of all cats are domestic (second premise), there must be some non-empty

intersection of those two groups of cats.[37] This shows that this argument is valid, but cannot be expressed in the traditional predicate logic with only the usual universal and existential quantifiers at our disposal. ♢

[37] We are using the existential presupposition — that cats exist — as well.

TO BE CONTINUED...

6 History Sketch

6.1 Timeline

The content of these notes follows logical progression which is not always the same as the chronological order of the invention of those ideas. For that reason, this section will provide a brief timeline of the development of logic, repeating some of the names and facts mentioned earlier.

Ἐπιμενίδης [Epimenides of Crete], 7th or 6th century BC He is the earliest semi-mythical character with a reference in a later source. Later Greek philosophers attributed the liar's paradox to him. There is no evidence he actually considered it himself — the paradox is based on a fragment of a verse attributed to him that hints at a contradiction. Reformulated for clarity, the paradox goes like this. Epimenides says: "Cretans always lie." But he is a Cretan himself. Is his sentence true or false? Three distinct ideas already emerge here. The first one is the idea of self-reference — the shadow of the uroboros. Then, this paradox already centers on the issue of a *statement* being *true or false*, thus anticipating the framework of propositional logic. Finally, in a somewhat implicit way, this paradox hints at the possibility of the truth and falsehood being decidable based on the *form* statement itself without any externalities brought to bear.

Σωκράτης [Socrates] (c.470–399 BC) He is the first historical figure whose name is inseparable from the history of logic.

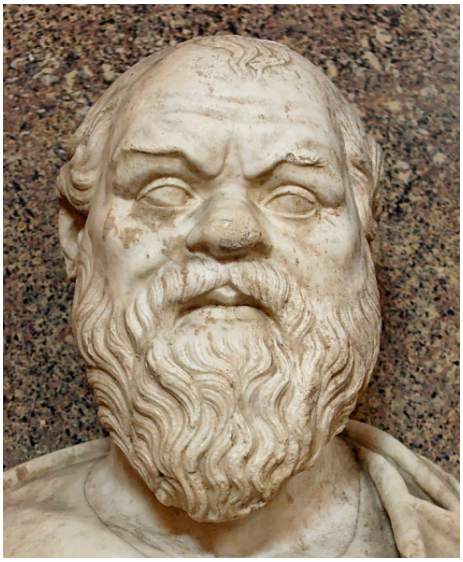


Figure 29: Σωκράτης [Socrates] (c.470–399 BC)

Athens of the fourth century BC had a form of direct democracy that placed a big value on the ability of people to formulate and articulate their ideas, and to use reasoning to convince others of their merit. That environment fostered the culture of public argument and necessitated the study of the general laws of argument we now call logic. Socratic school emerged against that background. While Socrates did not leave any books of his own, he founded a school and one of his students, Πλάτων [Plato] (c.429–c.347 BC), recorded some of the Socrates conversations in [31]. Socratic Dialogues brought rational reasoning in focus and made it a continuing theme in the development of culture.

Εὐκλείδης [Euclid of Megara] (c.435–c.365 BC) He was another pupil of Socrates (who reportedly was present at Socrates' death) Euclid founded the Megarian school of philosophy. The philosophers of that school already considered the liar's paradox, attributing it to Epimenides. Some of Euclid's successors developed logic to such an extent that they became a separate school, that became known as the Dialectical school. The work of the dialectical school on modal logic, logical conditionals, and propositional logic played an important role in the development of logic in antiquity.

Ἀριστοτέλης [Aristotle] (384–322 BC) [pg. 37] Aristotle was a student of Plato who established logic as an independent field. In his work

Organon [2], he fully developed the syllogistic logic, including the categories, predicates, and quantifiers.

Εὐκλείδης [Euclid of Alexandria] (c.325–c.265 BC) Euclid was a Greek mathematician who lived in Ptolemaic Egypt. He used `axiomatic method` in his study of geometry. His Στοιχεῖα [Elements] [11], a mathematical treatise consisting of 13 books, summarized all mathematics known at that time and became the standard for a rigorous treatment of any subject for the next millennia.

Χρύσιππος [Chrysippus of Soli] (c.279–c.204 BC) [pg. 27] Chrysippus was a student of Aristotle who succeeded him as the head of the Peripatetic School. He perfected the discipline of propositional logic, but only fragments of his works survive to this day [7].

Dissolution of the Roman empire resulted in the time of great upheaval in Europe, and many cultural treasures were lost. Fortunately, many of the ancient ideas and sources were preserved by the Islamic scholars, and reemerged in the medieval Europe around the turn of the first millennium. Medieval scholasticism placed a big emphasis on study of syllogisms.

Gottfried Wilhelm Leibnitz (1646–1716) [pg. 39] Leibniz was a German philosopher who co-invented, with Isaac Newton, the Mathematical Analysis. His approach was based on the concept of "monads" that represented `infinitesimals`. While intuitive and for this reason favored by physicists, the concept of infinitesimals looked problematic to generations of mathematicians that followed Leibnitz. The traditional foundation of analysis avoided infinitesimals and relied instead on the machinery of inequalities developed by Weierstrass and Cauchy. Only the new logical advances of Abraham Robinson around 1960 resolved these difficulties and restored infinitesimals to a fully legitimate status. In logic, he used what we call "Euler-Venn diagrams" to analyze syllogisms, and put forward, sometime after 1686, the idea of characteristica universalis. (That idea inspired Frege to create his Begriffsschrift.)

1847 — George Boole (1815–1864) [pg. 40], a self-taught British scientist, invents what we now call `Boolean algebra` and used it in [5] to study syllogisms with algebraic methods.

1873,74 — Georg Cantor (1845-1918), a German mathematician, outlined the basics of infinite set theory. His original theory suffered from the same problem as Begriffsschrift of Frege, invented just a few years later. However, his theory became "the garden of Eden" for mathematicians, providing both the framework for building all other mathematical concepts, and

a challenge and focus of efforts on the foundation of the subject. These efforts culminated in several axiomatic set theories.

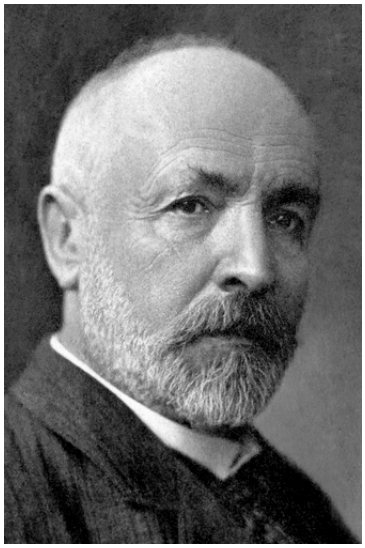


Figure 30: Georg Ferdinand Ludwig Philipp Cantor (1845–1918)

1879 — Friedrich Ludwig Gottlob Frege (1848–1925) [pg. 62], a German mathematician, invents [13] the "Begriffsschrift" and opens a new chapter in logic.

1889 — Giuseppe Peano (1858–1932), an Italian mathematician, publishes a logical definition of natural numbers (Peano axioms of arithmetic) in his book [30].

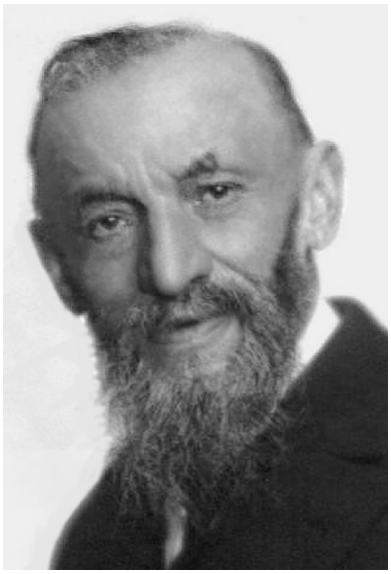


Figure 31: Giuseppe Peano (1858–1932)

1897 — Cesare Burali-Forti (1861–1931) an Italian mathematician, publishes a result [6] that (unknowingly to author) shows inconsistency of Cantor's set theory [38]. This result foreshadows the Russel's paradox that came 5 years later.

[38] That result is now known as the Burali-Forti's paradox. Assuming that the set $\mathcal{O}$ of all ordinal numbers existed, Burali-Forti proved that $\mathcal{O}$ must be well-ordered itself, and thus be its own member: $\mathcal{O} \in \mathcal{O}$, implying that $\mathcal{O}$ is smaller than $\mathcal{O}$ — which is a contradiction.

Figure 32: Cesare Burali-Forti (1861–1931)

1902 — Bertrand Arthur William Russell (1872–1970) [pg. 64], a British philosopher, sends a letter [36] to Frege which contains what is now known as the "Russell's paradox". Initiates the study of Mathematics foundations with Principia Mathematica.

David Hilbert (1862–1943) David Hilbert (1862–1943) and Wilhelm Ackermann (1896–1962). Grundzüge der theoretischen Logik (Principles of Mathematical Logic). Springer-Verlag efforts in logic [**?**]

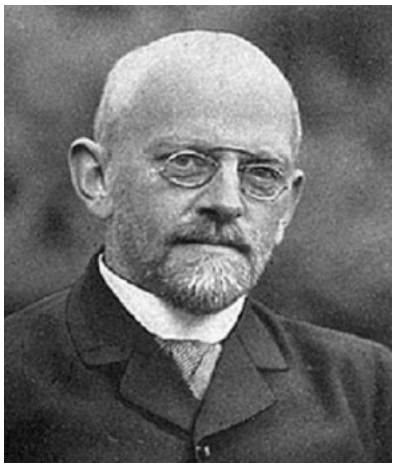


Figure 33: David Hilbert (1862–1943)

1924 — Моисей Эльевич Шейнфинкель [Moses Schönfinkel] (1889–1942), [pg. 52] a Soviet mathematician, student of Hilbert and a member of the Göttingen Logic School, invents `combinatory logic` as the framework for foundations of mathematics.

Jan Łukasiewicz (1878–1956)

Figure 34: Jan Leopold Łukasiewicz (1878–1956)

One of the founding fathers of the Lwów-Warsaw logic school.

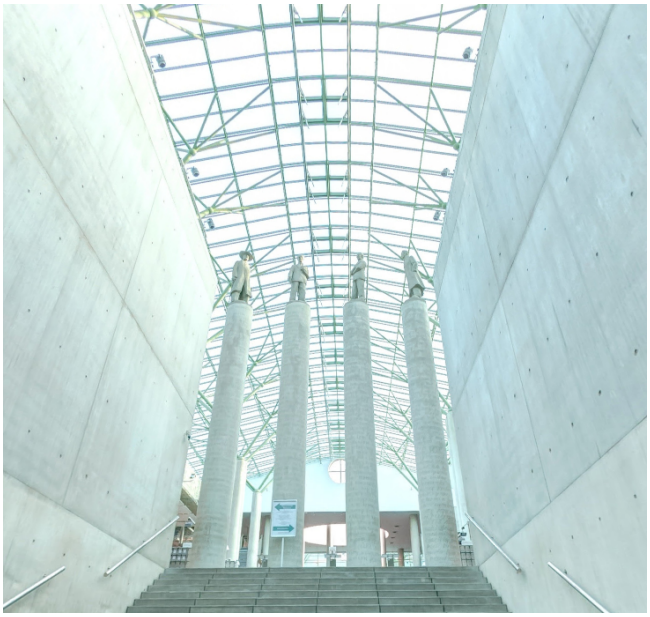


Figure 35: Kazimierz Twardowski, Jan Łukasiewicz, Alfred Tarski, and Stanisław Leśniewski - Warsaw University Library

In his 1926 seminars, made an observation that "real" mathematicians don't prove their theorems using the logical theories known at the time (including those by Łukasiewicz himself, Frege, and Hilbert). Poses the challenge to his colleagues to create a system that can be used in the real world.

1927 — Stanisław Jaśkowski (1906–1965) [pg. **??**], a Polish logician (and a student of Łukasiewicz) who accepted the challenge posed by his mentor, communicates his first (graphical) form of Natural Deduction at the First Polish Mathematical Congress [24].

1930 — Jacques Herbrand (1908–1931) [pg. **??**], a French mathematician, introduces Herbrand semantics in his thesis.

1930 — Haskell Brooks Curry (1900–1982) [pg. 52] publishes his paper [9] on combinatory logic.

1931 — Kurt Friedrich Gödel (1906–1978), at the time — an Austrian mathematician, publishes [16] the result now known as the "Gödel incompleteness theorem". His result shows the limits of formal methods and curbs Hilbert's hopes for axiomatization of mathematics.

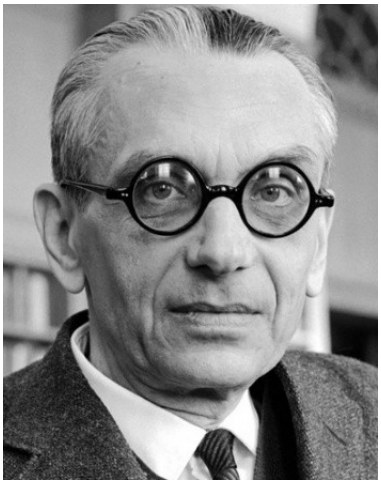


Figure 36: Kurt Friedrich Gödel (1906–1978)

1932 — Alonzo Church (1903–1995) [pg. 51], an American mathematician, invents the untyped lambda calculus [8].

1934 — Gerhard Gentzen (1909–1945) [pg. **??**] a German mathematician and a student of Hilbert, publishes [14] descriptions of several versions of Natural Deduction. Gentzen presents three different systems of deduction, including one for `intuitionistic logic`. Same year, 1934, Jaśkowski publishes his description of Natural Deduction, which is an independent effort from that of Gentzen [25], (See the comparison below.)

1935 — Stephen Cole Kleene (1909–1994), John Barkley Rosser, Sr. (1907–1989) [pg. 53], American mathematicians, present [26] what became to be known as the "Kleene-Rosser paradox", demonstrating inconsistency of logic model within combinatory and (untyped) lambda calculus.

??? — Haskell Brooks Curry (1900–1982) [pg. 52] simplifies Kleene-Rosser construction and presents the Curry's paradox showing inconsistency of any logic model within a system possessing a Y-combinator.

1936 — Alan Mathison Turing (1912–1954) describes the Universal Turing machine model of computation in [38]. This Universal Turing Machine provides an alternative model of computation to Church's lambda calculus, and while less suitable for human use, is immediately realizable in the physical world. It becomes the dominant model of computation until higher level programming languages start to take hold in 1950's.

Figure 37: Alan Mathison Turing (1912–1954)

1937 — Willard Van Orman Quine (1908–2000) an American philosopher, publishes [32] the "New Foundations" of the set theory.

1942 — John Barkley Rosser, Sr. (1907–1989) [pg. 53] an American mathematician, finds in [35] that Burali-Forti paradox applies to Quine's "New Foundations" necessitating a revision of that set theory (by Quine himself).

1952 — Frederic Brenton Fitch (1908–1987) [pg. **??**] an American logician, introduces his Natural Deduction notation in the textbook [12].

1959 — John McCarthy (1927–2011) [pg. 55] an American mathematician and computer scientist, invents the computer programming language Lisp by modeling it after Church's lambda calculus.

Abraham Robinson (1918–1974) Using model theory, Robinson was able to build a solid logical foundation for the classic — but held suspect for hundreds of years — infinitesimals-based approach to Mathematical Analysis. [33]

Figure 38: Abraham Robinson (1918–1974)

1965 — John Alan Robinson (1930–2016) a British-American mathematician, discovers the `resolution principle` and describes it in [34].

Figure 39: John Alan Robinson (1930–2016)

early 1970's — Robert Anthony Kowalski (1941–) a British-American mathematician, lays the theoretical foundations for the Prolog language, see e.g. [27]

Figure 40: Robert Anthony Kowalski (1941–)

6.2 Concluding Remarks

There are several different types of contributors and contributions in the above timeline. People who fostered the creation of new schools (Socrates, Aristotle, Hilbert, Łukasiewicz) not only advanced the subject itself, but gave cultural development an impulse that often persisted for many generations after them [39]. Masters of other fields who did not have logic as the main focus in all of their endeavors (Euclid of Alexandria, in some ways Leibnitz, Peano, to some extent Hilbert in his geometry works), — but wanted to *be* logical in their studies of other subjects: they moved the stake posts of logic into the new territory and filled the subject with the vital energy of its applications, giving the logicians that followed them the new spaces in which the subject could develop. There are those (Cantor, Frege, Schönfinkel, Hilbert, Curry, Church, Quine) who eagerly expanded the raw expressive power of logic — and those (Burali-Forti, Russell, Gödel, Kleene, Rosser) who pruned some of the wilder branches that ended up connecting truths and falsehoods, thus creating the short circuits of contradictions...

This unending cycle of generation and destruction, looped into a contradiction by its own self-reference, is driven by the conflict between the expansion of logic in its attempt to encompass the forever growing realms of human knowledge on the one hand, and taming of its power, forced by the need to guarantee the separation between the truth and the falsehood — on the other. While this cycle goes on, we are done — and we came back to our beginning.

[39] sometimes — as in the case of Aristotle — even *restarting* after a long hiatus.

Figure 41: The Uroboros

6.3 Appendix: Historical Examples of Different Notations

Consider a predicate logic argument:

$$\Big(\Big(\forall x : \Big(P(x) \Rightarrow Q(x)\Big)\Big) \wedge \Big(\exists x : \Big(\neg Q(x)\Big)\Big)\Big) \Rightarrow \Big(\exists x : \Big(\neg P(x)\Big)\Big)$$

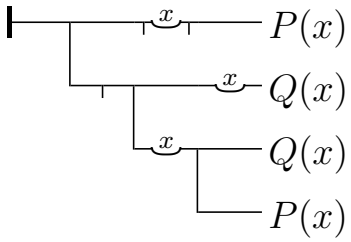
This argument can be presented in the style of Gerhard Gentzen which we used before already:

$$\forall x : \Big(P(x) \Rightarrow Q(x)\Big)$$

$$\exists x : \Big(\neg Q(x)\Big)$$

$$\exists x : \Big(\neg P(x)\Big)$$

Gentzen was the one who introduced the notations $\forall, \exists, \wedge$, and $\vee$, so it is not surprising that this form of writing looks very modern. The same argument would look completely differently in the amusingly idiosyncratic style of Friedrich Ludwig Gottlob Frege:



However, you may notice something familiar even here: the negation $\neg$ and the "turnstile" $\vdash$ that has been used ever since for an assertion of truth of some statement are the two symbols introduced by Frege which are still in use today.

Assuming now that we want to prove this argument, how would a natural deduction proof look in different notation systems?

1.		$\forall x : \Big(P(x) \Rightarrow Q(x)\Big)$	hypothesis
2.		$\exists x : \Big(\neg Q(x)\Big)$	hypothesis
3.	a	$\neg Q(a)$	sub-proof hypothesis
4.		$P(a) \Rightarrow Q(a)$	$\forall x$ elimination from 1 (taking $x = a$)
5.		$P(a)$	sub-sub-proof hypothesis
6.		$Q(a)$	$\Rightarrow$ elimination from 4, 5
7.		F	contradiction from 3, 6
8.		$\neg P(a)$	reduction ad absurdum from 5–7
9.		$\exists x : \Big(\neg P(x)\Big)$	$\exists$ introduction from 8
10.		$\exists x : \Big(\neg P(x)\Big)$	$\exists$ elimination from 2, 3–9

Figure 42: A Proof in the Style of Stanisław Jaśkowski

1. $\forall \mathsf{x} : \Big(\mathsf{P}(\mathsf{x}) \Rightarrow \mathsf{Q}(\mathsf{x})\Big)$	
2. $\exists \mathsf{x} : \Big(\neg \mathsf{Q}(\mathsf{x})\Big)$	
3. $\boxed{\mathsf{a}}\ \neg \mathsf{Q}(\mathsf{a})$	
4. $\mathsf{P}(\mathsf{a}) \Rightarrow \mathsf{Q}(\mathsf{a})$	$\forall$ **Elim:** 1
5. $\mathsf{P}(\mathsf{a})$	
6. $\mathsf{P}(\mathsf{a}) \Rightarrow \mathsf{Q}(\mathsf{a})$	**Reit:** 4
7. $\mathsf{P}(\mathsf{a})$	**Reit:** 5
8. $\mathsf{Q}(\mathsf{a})$	$\Rightarrow$ **Elim:** 6, 7
9. $\neg \mathsf{Q}(\mathsf{a})$	**Reit:** 3
9. $\bot$	$\bot$ **Intro:** 8, 9
10. $\neg \mathsf{P}(\mathsf{a})$	$\neg$ **Intro:** 5–9
11. $\exists \mathsf{x} : \Big(\neg \mathsf{P}(\mathsf{x})\Big)$	$\exists$ **Intro:** 10
12. $\exists \mathsf{x} : \Big(\neg \mathsf{P}(\mathsf{x})\Big)$	$\exists$ **Elim:** 2, 3–11

Figure 43: A Proof in the Style of Frederic Brenton Fitch

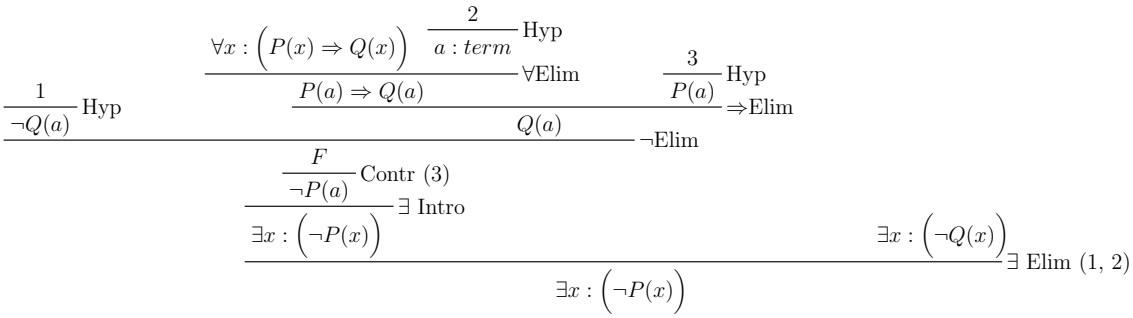


Figure 44: A Proof in the Style of Gerhard Gentzen

References

[1] The grounding of elementary number theory. [28], pages 266–273. in [28].

[2] Aristotle. *Ὄργανον ["Organon", The Instrument]*. 4th Century BC. URL: https://www.rbjones.com/rbjpub/philos/classics/aristotl/obook_draft.pdf.

[3] Jonathan Barnes. *Logical Matters: Essays in Ancient Philosophy II*. Oxford University Press, 2012. URL: https://www.wiko-berlin.de/fileadmin/Jahrbuchberichte/1984/1984_85_Barnes_Jonathan_Jahrbuchbericht.pdf.

[4] Michael Beaney, editor. *The Frege reader*. Blackwell Publishing, 1997. URL: https://dl1.cuni.cz/pluginfile.php/767005/mod_resource/content/0/Frege%20-%20Reader.pdf.

[5] George Boole. *The Mathematical Analysis of Logic*. Cambridge: MacMillan, Barclay, & MacMillan; London: George Bell., 1847. URL: https://www.gutenberg.org/files/36884/36884-pdf.pdf.

[6] Cesare Burali-Forti. Una questione sui numeri transfiniti. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 11:154–164, 1897. URL: https://zenodo.org/record/2362091/files/article.pdf.

[7] Chrysippus. *Logical Investigations (Surviving Fragments)*. In [3], 3rd Century BC. URL: https://www.wiko-berlin.de/fileadmin/Jahrbuchberichte/1984/1984_85_Barnes_Jonathan_Jahrbuchbericht.pdf.

[8] Alonzo Church. A set of postulates for the foundation of logic. *Annals of Mathematics*, 33(2):346–366, 1932. URL: https://raw.githubusercontent.com/emintham/Papers/master/Church-%20A%20Set%20of%20Postulates%20for%20the%20Foundation%20of%20Logic.pdf.

[9] Haskell Brooks Curry. Grundlagen der Kombinatorischen Logik [foundations of combinatorial logic]. *American Journal of Mathematics*, 52:509–536, 1930.

[10] Haskell Brooks Curry. The Inconsistency of Certain Formal Logics. *Journal of Symbolic Logic*, 7:115–117, 1942.

[11] Euclid. *Στοιχεῖα [The Elements]*. c.300 BC. URL: https://farside.ph.utexas.edu/books/Euclid/Elements.pdf.

[12] Frederic Brenton Fitch. *Symbolic Logic: An Introduction*. New York: The Ronald Press Company, 1952. URL: https://usa1lib.org/book/2725862/ae4b46.

[13] Gottlob Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens [Concept Script, a formal language of pure thought modelled upon that of arithmetic]*. Verlag von Louis Nebert, Halle, 1879. URL: https://gdz.sub.uni-goettingen.de/download/pdf/PPN538957069/PPN538957069.pdf, also a translation at: https://www.informationphilosopher.com/solutions/philosophers/frege/Frege_Begriffsschrift.pdf.

[14] Gerhard Gentzen. Untersuchungen über das logische Schließen [investigations into logical deduction], I and II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935. URL: https://pic.blog.plover.com/math/logic/lk/Gentzen1934.pdf, also a translation at: https://logic-teaching.github.io/prop/texts/Gentzen%201969%20-%20Investigations%20into%20Logical%20Deduction.pdf.

[15] Kurt Gödel. *Über die Vollständigkeit des Logikkalküls [On the Completeness of the Calculus of Logic]*. 1929.

[16] Kurt Friedrich Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I. *Monatshefte für Mathematik und Physik*, 38:173–98, 1931.

[17] David Hilbert. Les principes fondamentaux de la géométrie. *Annales scientifiques de l'École Normale Supérieure*, 17:103–209, 1900.

[18] David Hilbert. Axiomatisches denken. *Mathematische Annalen*, 78:405–415, 1917.

[19] David Hilbert. Die logischen grundlagen der mathematik. *Mathematische Annalen*, 88:151–165, 1923.

[20] David Hilbert. Über das unendliche. *Mathematische Annalen*, 95:161–190, 1926. URL: https://gdz.sub.uni-goettingen.de/download/pdf/PPN235181684_0095/PPN235181684_0095.pdf.

[21] David Hilbert. Probleme der grundlegung der mathematik. *Mathematische Annalen*, 102:1–9, 1930.

[22] David Hilbert. Die grundlegung der elementaren zahlenlehre. *Mathematische Annalen*, 104:485–494, 1931.

[23] Ackermann, Wilhelm Hilbert, David. *Grundzüge der theoretischen Logik [Principles of Mathematical Logic]*. Springer-Verlag, Berlin, 1928. URL: https://github.com/mdnahas/Peano_Book/blob/master/Peano.pdf.

[24] Stanisław Jaśkowski. Teoria dedukcji oparta na regułach założeniowych [theory of deduction based on suppositional rules]. *Księga pamiątkowa pierwszego polskiego zjazdu matematycznego, 1927 [Proceedings of the First Polish Mathematical Congress, 1927]*, page 36, 1929.

[25] Stanisław Jaśkowski. On the rules of suppositions in formal logic. *Studia Logica*, 1:5–32, 1934. URL: https://www.logik.ch/daten/jaskowski.pdf.

[26] Rosser, John Barkley, Sr. Kleene, Stephen Cole. The inconsistency of certain formal logics. *Annals of Mathematics*, 36:630–636, 1935.

[27] Robert A. Kowalski. Predicate logic as programming language. November 1973.

[28] Paolo Mancosu, editor. *From Brouwer to Hilbert: The debate on the foundations of mathematics in the 1920s*. Oxford University Press, New York, 1998.

[29] John McCarthy. Recursive Functions of Symbolic Expressions and their Computation by Machine. 1959.

[30] Guiseppe Peano. *Arithmetices Principia Nova Methodo Exposita [The principles of arithmetic presented by a new method]*. Turin: Bocca Brothers, 1889. URL: https://github.com/mdnahas/Peano_Book/blob/master/Peano.pdf.

[31] Plato. *Σωκρατικὸς λόγος [Socratic Dialogues]*. between 399 and 387 BC. URL: https://webs.ucm.es/info/diciex/gente/agf/plato/The_Dialogues_of_Plato_v0.1.pdf.

[32] Willard Van Orman Quine. New Foundations for Mathematical Logic. *The American Mathematical Monthly*, 44:70–80, 1937.

[33] Abraham Robinson. *Non-standard Analysis*. Princeton University Press, 1966.

[34] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Communications of the ACM*, 12:23–41, 1965. URL: https://dl.acm.org/doi/pdf/10.1145/321250.321253.

[35] Sr. Rosser, John Barkley. The Burali-Forti paradox. *Journal of Symbolic Logic*, 7:1–17, 1942.

[36] Bertrand Russell. Letter to Frege, of june 16, 1902, informing him of the contradiction in Grundgesetze der Arithmetik that has come to be known as Russell's paradox. [4]. URL: https://dl1.cuni.cz/pluginfile.php/767005/mod_resource/content/0/Frege%20-%20Reader.pdf.

[37] Moses Schönfinkel. Über die Bausteine der mathematischen Logik [on the building blocks of mathematical logic]. *Mathematische Annalen*, 92:305–316, 1924.

[38] Alan Mathison Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society (published 1937)*, 2:230–265, 1936.

[39] Ludwig Wittgenstein. Logisch-Philosophische Abhandlung. 1919. URL: https://www.wittgensteinproject.org/w/index.php?title=Logisch-philosophische_Abhandlung_(Darstellung_in_Baumform),.

[40] Ludwig Wittgenstein. Tractatus Logico-Philosophicus. 1922. URL: https://www.wittgensteinproject.org/w/index.php?title=Tractatus_Logico-Philosophicus_(English),.

Index