Contents

Chapter 1

First-order logic

First-order logic is a collection of formal systems used in mathematics, philosophy, linguistics, and computer science. It is also known as **first-order predicate calculus**, the **lower predicate calculus**, **quantification theory**, and **predicate logic**. First-order logic uses quantified variables over (non-logical) objects. It allows the use of sentences that contain variables, so that rather than propositions such as *Socrates is a man* one can have expressions in the form *X is a man* where X is a variable.[1] This distinguishes it from propositional logic, which does not use quantifiers.

A theory about a topic is usually a first-order logic together with a specified domain of discourse over which the quantified variables range, finitely many functions from that domain to itself, finitely many predicates defined on that domain, and a set of axioms believed to hold for those things. Sometimes "theory" is understood in a more formal sense, which is just a set of sentences in first-order logic.

The adjective "first-order" distinguishes first-order logic from higher-order logic in which there are predicates having predicates or functions as arguments, or in which one or both of predicate quantifiers or function quantifiers are permitted.[2] In first-order theories, predicates are often associated with sets. In interpreted higher-order theories, predicates may be interpreted as sets of sets.

There are many deductive systems for first-order logic which are both sound (all provable statements are true in all models) and complete (all statements which are true in all models are provable). Although the logical consequence relation is only semidecidable, much progress has been made in automated theorem proving in first-order logic. First-order logic also satisfies several metalogical theorems that make it amenable to analysis in proof theory, such as the Löwenheim–Skolem theorem and the compactness theorem.

First-order logic is the standard for the formalization of mathematics into axioms and is studied in the foundations of mathematics. Peano arithmetic and Zermelo–Fraenkel set theory are axiomatizations of number theory and set theory, respectively, into first-order logic. No first-order theory, however, has the strength to uniquely describe a structure with an infinite domain, such as the natural numbers or the real line. Axioms systems that do fully describe these two structures (that is, categorical axiom systems) can be obtained in stronger logics such as second-order logic.

For a history of first-order logic and how it came to dominate formal logic, see José Ferreirós (2001).

1.1 Introduction

While propositional logic deals with simple declarative propositions, first-order logic additionally covers predicates and quantification.

A predicate takes an entity or entities in the domain of discourse as input and outputs either True or False. Consider the two sentences "Socrates is a philosopher" and "Plato is a philosopher". In propositional logic, these sentences are viewed as being unrelated and might be denoted, for example, by variables such as p and q. The predicate "is a philosopher" occurs in both sentences, which have a common structure of "a is a philosopher". The variable a is instantiated as "Socrates" in the first sentence and is instantiated as "Plato" in the second sentence. The use of predicates, such as "is a philosopher" in this example, distinguishes first-order logic from propositional logic.

Relationships between predicates can be stated using logical connectives. Consider, for example, the first-order formula "if a is a philosopher, then a is a scholar". This formula is a conditional statement with "a is a philosopher" as its hypothesis and "a is a scholar" as its conclusion. The truth of this formula depends on which object is denoted

by a, and on the interpretations of the predicates "is a philosopher" and "is a scholar".

Quantifiers can be applied to variables in a formula. The variable a in the previous formula can be universally quantified, for instance, with the first-order sentence "For every a, if a is a philosopher, then a is a scholar". The universal quantifier "for every" in this sentence expresses the idea that the claim "if a is a philosopher, then a is a scholar" holds for *all* choices of a.

The *negation* of the sentence "For every a, if a is a philosopher, then a is a scholar" is logically equivalent to the sentence "There exists a such that a is a philosopher and a is not a scholar". The existential quantifier "there exists" expresses the idea that the claim "a is a philosopher and a is not a scholar" holds for *some* choice of a.

The predicates "is a philosopher" and "is a scholar" each take a single variable. In general, predicates can take several variables. In the first-order sentence "Socrates is the teacher of Plato", the predicate "is the teacher of" takes two variables.

An interpretation (or model) of a first-order formula specifies what each predicate means and the entities that can instantiate the variables. These entities form the domain of discourse or universe, which is usually required to be a nonempty set. For example, in interpretation with the domain of discourse consisting of all human beings and the predicate "is a philosopher" understood as "was the author of the *Republic*", the sentence "There exists a such that a is a philosopher" is seen as being true, as witnessed by Plato.

1.2 Syntax

There are two key parts of first-order logic. The syntax determines which collections of symbols are legal expressions in first-order logic, while the semantics determine the meanings behind these expressions.

1.2.1 Alphabet

Unlike natural languages, such as English, the language of first-order logic is completely formal, so that it can be mechanically determined whether a given expression is legal. There are two key types of legal expressions: **terms**, which intuitively represent objects, and **formulas**, which intuitively express predicates that can be true or false. The terms and formulas of first-order logic are strings of **symbols** which together form the **alphabet** of the language. As with all formal languages, the nature of the symbols themselves is outside the scope of formal logic; they are often regarded simply as letters and punctuation symbols.

It is common to divide the symbols of the alphabet into **logical symbols**, which always have the same meaning, and **non-logical symbols**, whose meaning varies by interpretation. For example, the logical symbol $\wedge$ always represents "and"; it is never interpreted as "or". On the other hand, a non-logical predicate symbol such as Phil(x) could be interpreted to mean "x is a philosopher", "x is a man named Philip", or any other unary predicate, depending on the interpretation at hand.

Logical symbols

There are several logical symbols in the alphabet, which vary by author but usually include:

- The quantifier symbols $\forall$ and $\exists$
- The logical connectives: $\wedge$ for conjunction, $\vee$ for disjunction, $\rightarrow$ for implication, $\leftrightarrow$ for biconditional, $\neg$ for negation. Occasionally other logical connective symbols are included. Some authors use Cpq, instead of $\rightarrow$, and Epq, instead of $\leftrightarrow$, especially in contexts where $\rightarrow$ is used for other purposes. Moreover, the horseshoe $\supset$ may replace $\rightarrow$; the triple-bar $\equiv$ may replace $\leftrightarrow$; a tilde (~), Np, or Fpq, may replace $\neg$; ||, or Apq may replace $\vee$; and &, Kpq, or the middle dot, $\cdot$, may replace $\wedge$, especially if these symbols are not available for technical reasons. (*Note*: the aforementioned symbols Cpq, Epq, Np, Apq, and Kpq are used in Polish notation.)
- Parentheses, brackets, and other punctuation symbols. The choice of such symbols varies depending on context.
- An infinite set of **variables**, often denoted by lowercase letters at the end of the alphabet x, y, z, ... Subscripts are often used to distinguish variables: x_0, x_1, x_2, ...
- An **equality symbol** (sometimes, **identity symbol**) =; see the section on equality below.

It should be noted that not all of these symbols are required – only one of the quantifiers, negation and conjunction, variables, brackets and equality suffice. There are numerous minor variations that may define additional logical symbols:

- Sometimes the truth constants T, Vpq, or $\top$, for "true" and F, Opq, or $\bot$, for "false" are included. Without any such logical operators of valence 0, these two constants can only be expressed using quantifiers.
- Sometimes additional logical connectives are included, such as the Sheffer stroke, Dpq (NAND), and exclusive or, Jpq.

Non-logical symbols

The non-logical symbols represent predicates (relations), functions and constants on the domain of discourse. It used to be standard practice to use a fixed, infinite set of non-logical symbols for all purposes. A more recent practice is to use different non-logical symbols according to the application one has in mind. Therefore, it has become necessary to name the set of all non-logical symbols used in a particular application. This choice is made via a **signature**.[3]

The traditional approach is to have only one, infinite, set of non-logical symbols (one signature) for all applications. Consequently, under the traditional approach there is only one language of first-order logic.[4] This approach is still common, especially in philosophically oriented books.

1. For every integer $n \geq 0$ there is a collection of n-**ary**, or n-**place**, **predicate symbols**. Because they represent relations between n elements, they are also called **relation symbols**. For each arity n we have an infinite supply of them:

   $$P^n{}_0, P^n{}_1, P^n{}_2, P^n{}_3, \ldots$$

2. For every integer $n \geq 0$ there are infinitely many n-ary **function symbols**:

   $$f^n{}_0, f^n{}_1, f^n{}_2, f^n{}_3, \ldots$$

In contemporary mathematical logic, the signature varies by application. Typical signatures in mathematics are {1, ×} or just {×} for groups, or {0, 1, +, ×, <} for ordered fields. There are no restrictions on the number of non-logical symbols. The signature can be empty, finite, or infinite, even uncountable. Uncountable signatures occur for example in modern proofs of the Löwenheim-Skolem theorem.

In this approach, every non-logical symbol is of one of the following types.

1. A **predicate symbol** (or **relation symbol**) with some **valence** (or **arity**, number of arguments) greater than or equal to 0. These are often denoted by uppercase letters P, Q, R,... .
   - Relations of valence 0 can be identified with propositional variables. For example, P, which can stand for any statement.
   - For example, $P(x)$ is a predicate variable of valence 1. One possible interpretation is "x is a man".
   - $Q(x,y)$ is a predicate variable of valence 2. Possible interpretations include "x is greater than y" and "x is the father of y".
2. A **function symbol**, with some valence greater than or equal to 0. These are often denoted by lowercase letters f, g, h,... .
   - Examples: $f(x)$ may be interpreted as for "the father of x". In arithmetic, it may stand for "-x". In set theory, it may stand for "the power set of x". In arithmetic, $g(x,y)$ may stand for "x+y". In set theory, it may stand for "the union of x and y".
   - Function symbols of valence 0 are called **constant symbols**, and are often denoted by lowercase letters at the beginning of the alphabet a, b, c,... . The symbol a may stand for Socrates. In arithmetic, it may stand for 0. In set theory, such a constant may stand for the empty set.

The traditional approach can be recovered in the modern approach by simply specifying the "custom" signature to consist of the traditional sequences of non-logical symbols.

1.2.2 Formation rules

The formation rules define the terms and formulas of first order logic. When terms and formulas are represented as strings of symbols, these rules can be used to write a formal grammar for terms and formulas. These rules are generally context-free (each production has a single symbol on the left side), except that the set of symbols may be allowed to be infinite and there may be many start symbols, for example the variables in the case of terms.

Terms

The set of **terms** is inductively defined by the following rules:

1. **Variables.** Any variable is a term.
2. **Functions.** Any expression $f(t_1,...,tn)$ of n arguments (where each argument ti is a term and f is a function symbol of valence n) is a term. In particular, symbols denoting individual constants are 0-ary function symbols, and are thus terms.

Only expressions which can be obtained by finitely many applications of rules 1 and 2 are terms. For example, no expression involving a predicate symbol is a term.

Formulas

The set of **formulas** (also called well-formed formulas [5] or **wff**s) is inductively defined by the following rules:

1. **Predicate symbols.** If P is an n-ary predicate symbol and $t1$, ..., tn are terms then $P(t_1,...,t_n)$ is a formula.
2. **Equality.** If the equality symbol is considered part of logic, and $t1$ and t_2 are terms, then $t_1 = t_2$ is a formula.
3. **Negation.** If φ is a formula, then $\neg \varphi$ is a formula.
4. **Binary connectives.** If φ and ψ are formulas, then $(\varphi \rightarrow \psi)$ is a formula. Similar rules apply to other binary logical connectives.
5. **Quantifiers.** If φ is a formula and x is a variable, then $\forall x \varphi$ (for all x, φ holds) and $\exists x \varphi$ (there exists x such that φ) are formulas.

Only expressions which can be obtained by finitely many applications of rules 1–5 are formulas. The formulas obtained from the first two rules are said to be **atomic formulas**.

For example,

$$\forall x \forall y (P(f(x)) \rightarrow \neg (P(x) \rightarrow Q(f(y), x, z)))$$

is a formula, if f is a unary function symbol, P a unary predicate symbol, and Q a ternary predicate symbol. On the other hand, $\forall x \, x \rightarrow$ is not a formula, although it is a string of symbols from the alphabet.

The role of the parentheses in the definition is to ensure that any formula can only be obtained in one way by following the inductive definition (in other words, there is a unique parse tree for each formula). This property is known as **unique readability** of formulas. There are many conventions for where parentheses are used in formulas. For example, some authors use colons or full stops instead of parentheses, or change the places in which parentheses are inserted. Each author's particular definition must be accompanied by a proof of unique readability.

This definition of a formula does not support defining an if-then-else function ite(c, a, b), where "c" is a condition expressed as a formula, that would return "a" if c is true, and "b" if it is false. This is because both predicates and functions can only accept terms as parameters, but the first parameter is a formula. Some languages built on first-order logic, such as SMT-LIB 2.0, add this.[6]

Notational conventions

For convenience, conventions have been developed about the precedence of the logical operators, to avoid the need to write parentheses in some cases. These rules are similar to the order of operations in arithmetic. A common convention is:

- $\neg$ is evaluated first
- $\wedge$ and $\vee$ are evaluated next
- Quantifiers are evaluated next
- $\rightarrow$ is evaluated last.

Moreover, extra punctuation not required by the definition may be inserted to make formulas easier to read. Thus the formula

$$(\neg \forall x P(x) \rightarrow \exists x \neg P(x))$$

might be written as

$$(\neg [\forall x P(x)]) \rightarrow \exists x [\neg P(x)].$$

In some fields, it is common to use infix notation for binary relations and functions, instead of the prefix notation defined above. For example, in arithmetic, one typically writes "2 + 2 = 4" instead of "=(+(2,2),4)". It is common to regard formulas in infix notation as abbreviations for the corresponding formulas in prefix notation.

The definitions above use infix notation for binary connectives such as $\rightarrow$. A less common convention is Polish notation, in which one writes $\rightarrow$, $\wedge$, and so on in front of their arguments rather than between them. This convention allows all punctuation symbols to be discarded. Polish notation is compact and elegant, but rarely used in practice because it is hard for humans to read it. In Polish notation, the formula

$$\forall x \forall y (P(f(x)) \rightarrow \neg (P(x) \rightarrow Q(f(y), x, z)))$$

becomes "∀x∀y→Pfx¬→ PxQfyxz".

1.2.3 Free and bound variables

Main article: Free variables and bound variables

In a formula, a variable may occur **free** or **bound**. Intuitively, a variable is free in a formula if it is not quantified: in $\forall y \, P(x, y)$, variable x is free while y is bound. The free and bound variables of a formula are defined inductively as follows.

1. **Atomic formulas.** If φ is an atomic formula then x is free in φ if and only if x occurs in φ. Moreover, there are no bound variables in any atomic formula.
2. **Negation.** x is free in $\neg \varphi$ if and only if x is free in φ. x is bound in $\neg \varphi$ if and only if x is bound in φ.
3. **Binary connectives.** x is free in $(\varphi \rightarrow \psi)$ if and only if x is free in either φ or ψ. x is bound in $(\varphi \rightarrow \psi)$ if and only if x is bound in either φ or ψ. The same rule applies to any other binary connective in place of $\rightarrow$.
4. **Quantifiers.** x is free in $\forall \, y \, \varphi$ if and only if x is free in φ and x is a different symbol from y. Also, x is bound in $\forall \, y \, \varphi$ if and only if x is y or x is bound in φ. The same rule holds with $\exists$ in place of $\forall$.

For example, in $\forall x \forall y (P(x) \rightarrow Q(x,f(x),z))$, x and y are bound variables, z is a free variable, and w is neither because it does not occur in the formula.

Free and bound variables of a formula need not be disjoint sets: x is both free and bound in $P(x) \rightarrow \forall x\, Q(x)$.

Freeness and boundness can be also specialized to specific occurrences of variables in a formula. For example, in $P(x) \rightarrow \forall x\, Q(x)$, the first occurrence of x is free while the second is bound. In other words, the x in $P(x)$ is free while the x in $\forall x\, Q(x)$ is bound.

A formula in first-order logic with no free variables is called a **first-order sentence**. These are the formulas that will have well-defined truth values under an interpretation. For example, whether a formula such as Phil(x) is true must depend on what x represents. But the sentence $\exists x$ Phil(x) will be either true or false in a given interpretation.

1.2.4 Examples

Ordered abelian groups

In mathematics the language of ordered abelian groups has one constant symbol 0, one unary function symbol −, one binary function symbol +, and one binary relation symbol ≤. Then:

- The expressions +(x, y) and +(x, +(y, −(z))) are **terms**. These are usually written as $x + y$ and $x + y − z$.
- The expressions +(x, y) = 0 and ≤(+(x, +(y, −(z))), +(x, y)) are **atomic formulas**.

  These are usually written as $x + y = 0$ and $x + y − z \leq x + y$.

- The expression $(\forall x \forall y \leq (+(x, y), z) \rightarrow \forall x \forall y + (x, y) = 0)$ is a **formula**, which is usually written as $\forall x \forall y (x + y \leq z) \rightarrow \forall x \forall y (x + y = 0)$.

Loving relation

English sentences like “everyone loves someone” can be formalized by first-order logic formulas like $\forall x \exists y\ L(x,y)$. This is accomplished by abbreviating the relation "x loves y" by $L(x,y)$. Using just the two quantifiers $\forall$ and $\exists$ and the loving relation symbol L, but no logical connectives and no function symbols (including constants), formulas with 8 different meanings can be built. The following diagrams show models for each of them, assuming that there are exactly five individuals a,...,e who can love (vertical axis) and be loved (horizontal axis). A small red box at row x and column y indicates $L(x,y)$. Only for the formulas 9 and 10 is the model unique, all other formulas may be satisfied by several models.

Each model, represented by a logical matrix, satisfies the formulas in its caption in a “minimal” way, i.e. whitening any red cell in any matrix would make it non-satisfying the corresponding formula. For example, formula 1 is also satisfied by the matrices at 3, 6, and 10, but not by those at 2, 4, 5, and 7. Conversely, the matrix shown at 6 satisfies 1, 2, 5, 6, 7, and 8, but not 3, 4, 9, and 10.

Some formulas imply others, i.e. *all* matrices satisfying the antecedent (LHS) also satisfy the conclusion (RHS) of the implication — e.g. formula 3 implies formula 1, i.e.: each matrix fulfilling formula 3 also fulfills formula 1, but not vice versa (see the Hasse diagram for this ordering relation). In contrast, only some matrices,[7] which satisfy formula 2, happen to satisfy also formula 5, whereas others,[8] also satisfying formula 2, do not; therefore formula 5 is not a logical consequence of formula 2.

The sequence of the quantifiers is important! So it is instructive to distinguish formulas 1: $\forall x\ \exists y\ L(y,x)$, and 3: $\exists x\ \forall y\ L(x,y)$. In both cases everyone is loved; but in the first case everyone (x) is loved by someone (y), in the second case everyone (y) is loved by just exactly one person (x).

1.3 Semantics

An interpretation of a first-order language assigns a denotation to all non-logical constants in that language. It also determines a domain of discourse that specifies the range of the quantifiers. The result is that each term is assigned an object that it represents, and each sentence is assigned a truth value. In this way, an interpretation provides semantic

meaning to the terms and formulas of the language. The study of the interpretations of formal languages is called formal semantics. What follows is a description of the standard or Tarskian semantics for first-order logic. (It is also possible to define game semantics for first-order logic, but aside from requiring the axiom of choice, game semantics agree with Tarskian semantics for first-order logic, so game semantics will not be elaborated herein.)

The domain of discourse *D* is a nonempty set of "objects" of some kind. Intuitively, a first-order formula is a statement about these objects; for example, $\exists x P(x)$ states the existence of an object x such that the predicate P is true where referred to it. The domain of discourse is the set of considered objects. For example, one can take D to be the set of integer numbers.

The interpretation of a function symbol is a function. For example, if the domain of discourse consists of integers, a function symbol f of arity 2 can be interpreted as the function that gives the sum of its arguments. In other words, the symbol f is associated with the function *I(f)* which, in this interpretation, is addition.

The interpretation of a constant symbol is a function from the one-element set D^0 to *D*, which can be simply identified with an object in *D*. For example, an interpretation may assign the value $I(c) = 10$ to the constant symbol c .

The interpretation of an *n*-ary predicate symbol is a set of *n*-tuples of elements of the domain of discourse. This means that, given an interpretation, a predicate symbol, and *n* elements of the domain of discourse, one can tell whether the predicate is true of those elements according to the given interpretation. For example, an interpretation *I(P)* of a binary predicate symbol *P* may be the set of pairs of integers such that the first one is less than the second. According to this interpretation, the predicate *P* would be true if its first argument is less than the second.

1.3.1 First-order structures

Main article: Structure (mathematical logic)

The most common way of specifying an interpretation (especially in mathematics) is to specify a **structure** (also called a **model**; see below). The structure consists of a nonempty set *D* that forms the domain of discourse and an interpretation *I* of the non-logical terms of the signature. This interpretation is itself a function:

- Each function symbol f of arity n is assigned a function *I(f)* from D^n to D . In particular, each constant symbol of the signature is assigned an individual in the domain of discourse.
- Each predicate symbol *P* of arity *n* is assigned a relation *I(P)* over D^n or, equivalently, a function from D^n to $\{\text{true}, \text{false}\}$. Thus each predicate symbol is interpreted by a Boolean-valued function on *D*.

1.3.2 Evaluation of truth values

A formula evaluates to true or false given an interpretation, and a **variable assignment** μ that associates an element of the domain of discourse with each variable. The reason that a variable assignment is required is to give meanings to formulas with free variables, such as $y = x$. The truth value of this formula changes depending on whether *x* and *y* denote the same individual.

First, the variable assignment μ can be extended to all terms of the language, with the result that each term maps to a single element of the domain of discourse. The following rules are used to make this assignment:

1. **Variables.** Each variable *x* evaluates to μ(*x*)
2. **Functions.** Given terms $t_1, \ldots, t_n$ that have been evaluated to elements $d_1, \ldots, d_n$ of the domain of discourse, and a *n*-ary function symbol f, the term $f(t_1, \ldots, t_n)$ evaluates to $(I(f))(d_1, \ldots, d_n)$.

Next, each formula is assigned a truth value. The inductive definition used to make this assignment is called the T-schema.

1. **Atomic formulas (1).** A formula $P(t_1, \ldots, t_n)$ is associated the value true or false depending on whether $\langle v_1, \ldots, v_n \rangle \in I(P)$, where $v_1, \ldots, v_n$ are the evaluation of the terms $t_1, \ldots, t_n$ and $I(P)$ is the interpretation of P , which by assumption is a subset of D^n .

2. **Atomic formulas (2).** A formula $t_1 = t_2$ is assigned true if t_1 and t_2 evaluate to the same object of the domain of discourse (see the section on equality below).
3. **Logical connectives.** A formula in the form $\neg\phi$, $\phi \rightarrow \psi$, etc. is evaluated according to the truth table for the connective in question, as in propositional logic.
4. **Existential quantifiers.** A formula $\exists x\phi(x)$ is true according to *M* and μ if there exists an evaluation μ' of the variables that only differs from μ regarding the evaluation of *x* and such that φ is true according to the interpretation *M* and the variable assignment μ' . This formal definition captures the idea that $\exists x\phi(x)$ is true if and only if there is a way to choose a value for *x* such that φ(*x*) is satisfied.
5. **Universal quantifiers.** A formula $\forall x\phi(x)$ is true according to *M* and μ if φ(*x*) is true for every pair composed by the interpretation *M* and some variable assignment μ' that differs from μ only on the value of *x*. This captures the idea that $\forall x\phi(x)$ is true if every possible choice of a value for *x* causes φ(*x*) to be true.

If a formula does not contain free variables, and so is a sentence, then the initial variable assignment does not affect its truth value. In other words, a sentence is true according to *M* and μ if and only if it is true according to *M* and every other variable assignment μ' .

There is a second common approach to defining truth values that does not rely on variable assignment functions. Instead, given an interpretation *M*, one first adds to the signature a collection of constant symbols, one for each element of the domain of discourse in *M*; say that for each *d* in the domain the constant symbol *cd* is fixed. The interpretation is extended so that each new constant symbol is assigned to its corresponding element of the domain. One now defines truth for quantified formulas syntactically, as follows:

1. **Existential quantifiers (alternate).** A formula $\exists x\phi(x)$ is true according to *M* if there is some *d* in the domain of discourse such that $\phi(c_d)$ holds. Here $\phi(c_d)$ is the result of substituting *cd* for every free occurrence of *x* in φ.
2. **Universal quantifiers (alternate).** A formula $\forall x\phi(x)$ is true according to *M* if, for every *d* in the domain of discourse, $\phi(c_d)$ is true according to *M*.

This alternate approach gives exactly the same truth values to all sentences as the approach via variable assignments.

1.3.3 Validity, satisfiability, and logical consequence

See also: Satisfiability

If a sentence φ evaluates to True under a given interpretation *M*, one says that *M* **satisfies** φ; this is denoted $M \vDash \varphi$. A sentence is **satisfiable** if there is some interpretation under which it is true.

Satisfiability of formulas with free variables is more complicated, because an interpretation on its own does not determine the truth value of such a formula. The most common convention is that a formula with free variables is said to be satisfied by an interpretation if the formula remains true regardless which individuals from the domain of discourse are assigned to its free variables. This has the same effect as saying that a formula is satisfied if and only if its universal closure is satisfied.

A formula is **logically valid** (or simply **valid**) if it is true in every interpretation. These formulas play a role similar to tautologies in propositional logic.

A formula φ is a **logical consequence** of a formula ψ if every interpretation that makes ψ true also makes φ true. In this case one says that φ is logically implied by ψ.

1.3.4 Algebraizations

An alternate approach to the semantics of first-order logic proceeds via abstract algebra. This approach generalizes the Lindenbaum–Tarski algebras of propositional logic. There are three ways of eliminating quantified variables from first-order logic that do not involve replacing quantifiers with other variable binding term operators:

- Cylindric algebra, by Alfred Tarski and his coworkers;

- Polyadic algebra, by Paul Halmos;
- Predicate functor logic, mainly due to Willard Quine.

These algebras are all lattices that properly extend the two-element Boolean algebra.

Tarski and Givant (1987) showed that the fragment of first-order logic that has no atomic sentence lying in the scope of more than three quantifiers has the same expressive power as relation algebra. This fragment is of great interest because it suffices for Peano arithmetic and most axiomatic set theory, including the canonical ZFC. They also prove that first-order logic with a primitive ordered pair is equivalent to a relation algebra with two ordered pair projection functions.

1.3.5 First-order theories, models, and elementary classes

A **first-order theory** of a particular signature is a set of axioms, which are sentences consisting of symbols from that signature. The set of axioms is often finite or recursively enumerable, in which case the theory is called **effective**. Some authors require theories to also include all logical consequences of the axioms. The axioms are considered to hold within the theory and from them other sentences that hold within the theory can be derived.

A first-order structure that satisfies all sentences in a given theory is said to be a **model** of the theory. An **elementary class** is the set of all structures satisfying a particular theory. These classes are a main subject of study in model theory.

Many theories have an **intended interpretation**, a certain model that is kept in mind when studying the theory. For example, the intended interpretation of Peano arithmetic consists of the usual natural numbers with their usual operations. However, the Löwenheim–Skolem theorem shows that most first-order theories will also have other, nonstandard models.

A theory is **consistent** if it is not possible to prove a contradiction from the axioms of the theory. A theory is **complete** if, for every formula in its signature, either that formula or its negation is a logical consequence of the axioms of the theory. Gödel's incompleteness theorem shows that effective first-order theories that include a sufficient portion of the theory of the natural numbers can never be both consistent and complete.

For more information on this subject see List of first-order theories and Theory (mathematical logic)

1.3.6 Empty domains

Main article: Empty domain

The definition above requires that the domain of discourse of any interpretation must be a nonempty set. There are settings, such as inclusive logic, where empty domains are permitted. Moreover, if a class of algebraic structures includes an empty structure (for example, there is an empty poset), that class can only be an elementary class in first-order logic if empty domains are permitted or the empty structure is removed from the class.

There are several difficulties with empty domains, however:

- Many common rules of inference are only valid when the domain of discourse is required to be nonempty. One example is the rule stating that $\phi \vee \exists x \psi$ implies $\exists x(\phi \vee \psi)$ when x is not a free variable in φ. This rule, which is used to put formulas into prenex normal form, is sound in nonempty domains, but unsound if the empty domain is permitted.
- The definition of truth in an interpretation that uses a variable assignment function cannot work with empty domains, because there are no variable assignment functions whose range is empty. (Similarly, one cannot assign interpretations to constant symbols.) This truth definition requires that one must select a variable assignment function (μ above) before truth values for even atomic formulas can be defined. Then the truth value of a sentence is defined to be its truth value under any variable assignment, and it is proved that this truth value does not depend on which assignment is chosen. This technique does not work if there are no assignment functions at all; it must be changed to accommodate empty domains.

Thus, when the empty domain is permitted, it must often be treated as a special case. Most authors, however, simply exclude the empty domain by definition.

1.4 Deductive systems

A **deductive system** is used to demonstrate, on a purely syntactic basis, that one formula is a logical consequence of another formula. There are many such systems for first-order logic, including Hilbert-style deductive systems, natural deduction, the sequent calculus, the tableaux method, and resolution. These share the common property that a deduction is a finite syntactic object; the format of this object, and the way it is constructed, vary widely. These finite deductions themselves are often called **derivations** in proof theory. They are also often called proofs, but are completely formalized unlike natural-language mathematical proofs.

A deductive system is **sound** if any formula that can be derived in the system is logically valid. Conversely, a deductive system is **complete** if every logically valid formula is derivable. All of the systems discussed in this article are both sound and complete. They also share the property that it is possible to effectively verify that a purportedly valid deduction is actually a deduction; such deduction systems are called **effective**.

A key property of deductive systems is that they are purely syntactic, so that derivations can be verified without considering any interpretation. Thus a sound argument is correct in every possible interpretation of the language, regardless whether that interpretation is about mathematics, economics, or some other area.

In general, logical consequence in first-order logic is only semidecidable: if a sentence A logically implies a sentence B then this can be discovered (for example, by searching for a proof until one is found, using some effective, sound, complete proof system). However, if A does not logically imply B, this does not mean that A logically implies the negation of B. There is no effective procedure that, given formulas A and B, always correctly decides whether A logically implies B.

1.4.1 Rules of inference

Further information: List of rules of inference

A **rule of inference** states that, given a particular formula (or set of formulas) with a certain property as a hypothesis, another specific formula (or set of formulas) can be derived as a conclusion. The rule is sound (or truth-preserving) if it preserves validity in the sense that whenever any interpretation satisfies the hypothesis, that interpretation also satisfies the conclusion.

For example, one common rule of inference is the **rule of substitution**. If t is a term and φ is a formula possibly containing the variable x, then $\varphi[t/x]$ is the result of replacing all free instances of x by t in φ. The substitution rule states that for any φ and any term t, one can conclude $\varphi[t/x]$ from φ provided that no free variable of t becomes bound during the substitution process. (If some free variable of t becomes bound, then to substitute t for x it is first necessary to change the bound variables of φ to differ from the free variables of t.)

To see why the restriction on bound variables is necessary, consider the logically valid formula φ given by $\exists x(x = y)$, in the signature of (0,1,+,×,=) of arithmetic. If t is the term "x + 1", the formula $\varphi[t/y]$ is $\exists x(x = x+1)$, which will be false in many interpretations. The problem is that the free variable x of t became bound during the substitution. The intended replacement can be obtained by renaming the bound variable x of φ to something else, say z, so that the formula after substitution is $\exists z(z = x + 1)$, which is again logically valid.

The substitution rule demonstrates several common aspects of rules of inference. It is entirely syntactical; one can tell whether it was correctly applied without appeal to any interpretation. It has (syntactically defined) limitations on when it can be applied, which must be respected to preserve the correctness of derivations. Moreover, as is often the case, these limitations are necessary because of interactions between free and bound variables that occur during syntactic manipulations of the formulas involved in the inference rule.

1.4.2 Hilbert-style systems and natural deduction

A deduction in a Hilbert-style deductive system is a list of formulas, each of which is a **logical axiom**, a hypothesis that has been assumed for the derivation at hand, or follows from previous formulas via a rule of inference. The logical axioms consist of several axiom schemas of logically valid formulas; these encompass a significant amount of propositional logic. The rules of inference enable the manipulation of quantifiers. Typical Hilbert-style systems have a small number of rules of inference, along with several infinite schemas of logical axioms. It is common to have only modus ponens and universal generalization as rules of inference.

Natural deduction systems resemble Hilbert-style systems in that a deduction is a finite list of formulas. However, natural deduction systems have no logical axioms; they compensate by adding additional rules of inference that can be used to manipulate the logical connectives in formulas in the proof.

1.4.3 Sequent calculus

Further information: Sequent calculus

The sequent calculus was developed to study the properties of natural deduction systems. Instead of working with one formula at a time, it uses **sequents**, which are expressions of the form

$$A_1, \ldots, A_n \vdash B_1, \ldots, B_k,$$

where A_1, ..., An, B_1, ..., Bk are formulas and the turnstile symbol $\vdash$ is used as punctuation to separate the two halves. Intuitively, a sequent expresses the idea that $(A_1 \wedge \cdots \wedge A_n)$ implies $(B_1 \vee \cdots \vee B_k)$.

1.4.4 Tableaux method

Further information: Method of analytic tableaux

Unlike the methods just described, the derivations in the tableaux method are not lists of formulas. Instead, a derivation is a tree of formulas. To show that a formula A is provable, the tableaux method attempts to demonstrate that the negation of A is unsatisfiable. The tree of the derivation has $\neg A$ at its root; the tree branches in a way that reflects the structure of the formula. For example, to show that $C \vee D$ is unsatisfiable requires showing that C and D are each unsatisfiable; this corresponds to a branching point in the tree with parent $C \vee D$ and children C and D.

1.4.5 Resolution

The resolution rule is a single rule of inference that, together with unification, is sound and complete for first-order logic. As with the tableaux method, a formula is proved by showing that the negation of the formula is unsatisfiable. Resolution is commonly used in automated theorem proving.

The resolution method works only with formulas that are disjunctions of atomic formulas; arbitrary formulas must first be converted to this form through Skolemization. The resolution rule states that from the hypotheses $A_1 \vee \cdots \vee A_k \vee C$ and $B_1 \vee \cdots \vee B_l \vee \neg C$, the conclusion $A_1 \vee \cdots \vee A_k \vee B_1 \vee \cdots \vee B_l$ can be obtained.

1.4.6 Provable identities

The following sentences can be called "identities" because the main connective in each is the biconditional.

$\neg \forall x \, P(x) \Leftrightarrow \exists x \, \neg P(x)$

$\neg \exists x \, P(x) \Leftrightarrow \forall x \, \neg P(x)$

$\forall x \, \forall y \, P(x, y) \Leftrightarrow \forall y \, \forall x \, P(x, y)$

$\exists x \, \exists y \, P(x, y) \Leftrightarrow \exists y \, \exists x \, P(x, y)$

$\forall x \, P(x) \wedge \forall x \, Q(x) \Leftrightarrow \forall x \, (P(x) \wedge Q(x))$

$\exists x \, P(x) \vee \exists x \, Q(x) \Leftrightarrow \exists x \, (P(x) \vee Q(x))$

$P \wedge \exists x \, Q(x) \Leftrightarrow \exists x \, (P \wedge Q(x))$ (where x must not occur free in P)

$P \vee \forall x \, Q(x) \Leftrightarrow \forall x \, (P \vee Q(x))$ (where x must not occur free in P)

1.5 Equality and its axioms

There are several different conventions for using equality (or identity) in first-order logic. The most common convention, known as **first-order logic with equality**, includes the equality symbol as a primitive logical symbol which is always interpreted as the real equality relation between members of the domain of discourse, such that the "two" given members are the same member. This approach also adds certain axioms about equality to the deductive system employed. These equality axioms are:

1. **Reflexivity**. For each variable x, $x = x$.
2. **Substitution for functions.** For all variables x and y, and any function symbol f,

   $$x = y \rightarrow f(...,x,...) = f(...,y,...).$$

3. **Substitution for formulas**. For any variables x and y and any formula $\varphi(x)$, if φ' is obtained by replacing any number of free occurrences of x in φ with y, such that these remain free occurrences of y, then

   $$x = y \rightarrow (\varphi \rightarrow \varphi').$$

These are axiom schemas, each of which specifies an infinite set of axioms. The third schema is known as **Leibniz's law**, "the principle of substitutivity", "the indiscernibility of identicals", or "the replacement property". The second schema, involving the function symbol f, is (equivalent to) a special case of the third schema, using the formula

$$x = y \rightarrow (f(...,x,...) = z \rightarrow f(...,y,...) = z).$$

Many other properties of equality are consequences of the axioms above, for example:

1. **Symmetry.** If $x = y$ then $y = x$.
2. **Transitivity.** If $x = y$ and $y = z$ then $x = z$.

1.5.1 First-order logic without equality

An alternate approach considers the equality relation to be a non-logical symbol. This convention is known as **first-order logic without equality**. If an equality relation is included in the signature, the axioms of equality must now be added to the theories under consideration, if desired, instead of being considered rules of logic. The main difference between this method and first-order logic with equality is that an interpretation may now interpret two distinct individuals as "equal" (although, by Leibniz's law, these will satisfy exactly the same formulas under any interpretation). That is, the equality relation may now be interpreted by an arbitrary equivalence relation on the domain of discourse that is congruent with respect to the functions and relations of the interpretation.

When this second convention is followed, the term **normal model** is used to refer to an interpretation where no distinct individuals a and b satisfy $a = b$. In first-order logic with equality, only normal models are considered, and so there is no term for a model other than a normal model. When first-order logic without equality is studied, it is necessary to amend the statements of results such as the Löwenheim–Skolem theorem so that only normal models are considered.

First-order logic without equality is often employed in the context of second-order arithmetic and other higher-order theories of arithmetic, where the equality relation between sets of natural numbers is usually omitted.

1.5.2 Defining equality within a theory

If a theory has a binary formula $A(x,y)$ which satisfies reflexivity and Leibniz's law, the theory is said to have equality, or to be a theory with equality. The theory may not have all instances of the above schemas as axioms, but rather as derivable theorems. For example, in theories with no function symbols and a finite number of relations, it is possible to define equality in terms of the relations, by defining the two terms s and t to be equal if any relation is unchanged by changing s to t in any argument.

Some theories allow other *ad hoc* definitions of equality:

- In the theory of partial orders with one relation symbol ≤, one could define $s = t$ to be an abbreviation for $s \leq t \wedge t \leq s$.

- In set theory with one relation $\in$, one may define $s = t$ to be an abbreviation for $\forall x\ (s \in x \leftrightarrow t \in x) \wedge \forall x\ (x \in s \leftrightarrow x \in t)$. This definition of equality then automatically satisfies the axioms for equality. In this case, one should replace the usual axiom of extensionality, $\forall x \forall y[\forall z(z \in x \Leftrightarrow z \in y) \Rightarrow x = y]$, by $\forall x \forall y[\forall z(z \in x \Leftrightarrow z \in y) \Rightarrow \forall z(x \in z \Leftrightarrow y \in z)]$, i.e. if x and y have the same elements, then they belong to the same sets.

1.6 Metalogical properties

One motivation for the use of first-order logic, rather than higher-order logic, is that first-order logic has many metalogical properties that stronger logics do not have. These results concern general properties of first-order logic itself, rather than properties of individual theories. They provide fundamental tools for the construction of models of first-order theories.

1.6.1 Completeness and undecidability

Gödel's completeness theorem, proved by Kurt Gödel in 1929, establishes that there are sound, complete, effective deductive systems for first-order logic, and thus the first-order logical consequence relation is captured by finite provability. Naively, the statement that a formula φ logically implies a formula ψ depends on every model of φ; these models will in general be of arbitrarily large cardinality, and so logical consequence cannot be effectively verified by checking every model. However, it is possible to enumerate all finite derivations and search for a derivation of ψ from φ. If ψ is logically implied by φ, such a derivation will eventually be found. Thus first-order logical consequence is semidecidable: it is possible to make an effective enumeration of all pairs of sentences (φ,ψ) such that ψ is a logical consequence of φ.

Unlike propositional logic, first-order logic is undecidable (although semidecidable), provided that the language has at least one predicate of arity at least 2 (other than equality). This means that there is no decision procedure that determines whether arbitrary formulas are logically valid. This result was established independently by Alonzo Church and Alan Turing in 1936 and 1937, respectively, giving a negative answer to the Entscheidungsproblem posed by David Hilbert in 1928. Their proofs demonstrate a connection between the unsolvability of the decision problem for first-order logic and the unsolvability of the halting problem.

There are systems weaker than full first-order logic for which the logical consequence relation is decidable. These include propositional logic and monadic predicate logic, which is first-order logic restricted to unary predicate symbols and no function symbols. Other logics with no function symbols which are decidable are the guarded fragment of first-order logic, as well as two-variable logic. The Bernays–Schönfinkel class of first-order formulas is also decidable. Decidable subsets of first-order logic are also studied in the framework of description logics.

1.6.2 The Löwenheim–Skolem theorem

The Löwenheim–Skolem theorem shows that if a first-order theory of cardinality λ has an infinite model, then it has models of every infinite cardinality greater than or equal to λ. One of the earliest results in model theory, it implies that it is not possible to characterize countability or uncountability in a first-order language. That is, there is no first-order formula φ(x) such that an arbitrary structure M satisfies φ if and only if the domain of discourse of M is countable (or, in the second case, uncountable).

The Löwenheim–Skolem theorem implies that infinite structures cannot be categorically axiomatized in first-order logic. For example, there is no first-order theory whose only model is the real line: any first-order theory with an infinite model also has a model of cardinality larger than the continuum. Since the real line is infinite, any theory satisfied by the real line is also satisfied by some nonstandard models. When the Löwenheim–Skolem theorem is applied to first-order set theories, the nonintuitive consequences are known as Skolem's paradox.

1.6.3 The compactness theorem

The compactness theorem states that a set of first-order sentences has a model if and only if every finite subset of it has a model. This implies that if a formula is a logical consequence of an infinite set of first-order axioms, then it is a logical consequence of some finite number of those axioms. This theorem was proved first by Kurt Gödel as a consequence of the completeness theorem, but many additional proofs have been obtained over time. It is a central tool in model theory, providing a fundamental method for constructing models.

The compactness theorem has a limiting effect on which collections of first-order structures are elementary classes. For example, the compactness theorem implies that any theory that has arbitrarily large finite models has an infinite model. Thus the class of all finite graphs is not an elementary class (the same holds for many other algebraic structures).

There are also more subtle limitations of first-order logic that are implied by the compactness theorem. For example, in computer science, many situations can be modeled as a directed graph of states (nodes) and connections (directed edges). Validating such a system may require showing that no "bad" state can be reached from any "good" state. Thus one seeks to determine if the good and bad states are in different connected components of the graph. However, the compactness theorem can be used to show that connected graphs are not an elementary class in first-order logic, and there is no formula $\varphi(x,y)$ of first-order logic, in the logic of graphs, that expresses the idea that there is a path from x to y. Connectedness can be expressed in second-order logic, however, but not with only existential set quantifiers, as Σ^1_1 also enjoys compactness.

1.6.4 Lindström's theorem

Main article: Lindström's theorem

Per Lindström showed that the metalogical properties just discussed actually characterize first-order logic in the sense that no stronger logic can also have those properties (Ebbinghaus and Flum 1994, Chapter XIII). Lindström defined a class of abstract logical systems, and a rigorous definition of the relative strength of a member of this class. He established two theorems for systems of this type:

- A logical system satisfying Lindström's definition that contains first-order logic and satisfies both the Löwenheim–Skolem theorem and the compactness theorem must be equivalent to first-order logic.
- A logical system satisfying Lindström's definition that has a semidecidable logical consequence relation and satisfies the Löwenheim–Skolem theorem must be equivalent to first-order logic.

1.7 Limitations

Although first-order logic is sufficient for formalizing much of mathematics, and is commonly used in computer science and other fields, it has certain limitations. These include limitations on its expressiveness and limitations of the fragments of natural languages that it can describe.

For instance, first-order logic is undecidable, meaning a sound, complete and terminating decision algorithm is impossible. This has led to the study of interesting decidable fragments such as C_2, first-order logic with two variables and the counting quantifiers $\exists^{\geq n}$ and $\exists^{\leq n}$ (these quantifiers are, respectively, "there exists at least n" and "there exists at most n") (Horrocks 2010).

1.7.1 Expressiveness

The Löwenheim–Skolem theorem shows that if a first-order theory has any infinite model, then it has infinite models of every cardinality. In particular, no first-order theory with an infinite model can be categorical. Thus there is no first-order theory whose only model has the set of natural numbers as its domain, or whose only model has the set of real numbers as its domain. Many extensions of first-order logic, including infinitary logics and higher-order logics, are more expressive in the sense that they do permit categorical axiomatizations of the natural numbers or real numbers. This expressiveness comes at a metalogical cost, however: by Lindström's theorem, the compactness theorem and the downward Löwenheim–Skolem theorem cannot hold in any logic stronger than first-order.

1.7.2 Formalizing natural languages

First-order logic is able to formalize many simple quantifier constructions in natural language, such as "every person who lives in Perth lives in Australia". But there are many more complicated features of natural language that cannot be expressed in (single-sorted) first-order logic. "Any logical system which is appropriate as an instrument for the analysis of natural language needs a much richer structure than first-order predicate logic" (Gamut 1991, p. 75).

1.8 Restrictions, extensions, and variations

There are many variations of first-order logic. Some of these are inessential in the sense that they merely change notation without affecting the semantics. Others change the expressive power more significantly, by extending the semantics through additional quantifiers or other new logical symbols. For example, infinitary logics permit formulas of infinite size, and modal logics add symbols for possibility and necessity.

1.8.1 Restricted languages

First-order logic can be studied in languages with fewer logical symbols than were described above.

- Because $\exists x \phi(x)$ can be expressed as $\neg \forall x \neg \phi(x)$, and $\forall x \phi(x)$ can be expressed as $\neg \exists x \neg \phi(x)$, either of the two quantifiers $\exists$ and $\forall$ can be dropped.
- Since $\phi \vee \psi$ can be expressed as $\neg(\neg\phi \wedge \neg\psi)$ and $\phi \wedge \psi$ can be expressed as $\neg(\neg\phi \vee \neg\psi)$, either $\vee$ or $\wedge$ can be dropped. In other words, it is sufficient to have $\neg$ and $\vee$, or $\neg$ and $\wedge$, as the only logical connectives.
- Similarly, it is sufficient to have only $\neg$ and $\rightarrow$ as logical connectives, or to have only the Sheffer stroke (NAND) or the Peirce arrow (NOR) operator.
- It is possible to entirely avoid function symbols and constant symbols, rewriting them via predicate symbols in an appropriate way. For example, instead of using a constant symbol 0 one may use a predicate $0(x)$ (interpreted as $x = 0$), and replace every predicate such as $P(0, y)$ with $\forall x \ (0(x) \rightarrow P(x, y))$. A function such as $f(x_1, x_2, ..., x_n)$ will similarly be replaced by a predicate $F(x_1, x_2, ..., x_n, y)$ interpreted as $y = f(x_1, x_2, ..., x_n)$. This change requires adding additional axioms to the theory at hand, so that interpretations of the predicate symbols used have the correct semantics.

Restrictions such as these are useful as a technique to reduce the number of inference rules or axiom schemas in deductive systems, which leads to shorter proofs of metalogical results. The cost of the restrictions is that it becomes more difficult to express natural-language statements in the formal system at hand, because the logical connectives used in the natural language statements must be replaced by their (longer) definitions in terms of the restricted collection of logical connectives. Similarly, derivations in the limited systems may be longer than derivations in systems that include additional connectives. There is thus a trade-off between the ease of working within the formal system and the ease of proving results about the formal system.

It is also possible to restrict the arities of function symbols and predicate symbols, in sufficiently expressive theories. One can in principle dispense entirely with functions of arity greater than 2 and predicates of arity greater than 1 in theories that include a pairing function. This is a function of arity 2 that takes pairs of elements of the domain and returns an ordered pair containing them. It is also sufficient to have two predicate symbols of arity 2 that define projection functions from an ordered pair to its components. In either case it is necessary that the natural axioms for a pairing function and its projections are satisfied.

1.8.2 Many-sorted logic

Ordinary first-order interpretations have a single domain of discourse over which all quantifiers range. **Many-sorted first-order logic** allows variables to have different **sorts**, which have different domains. This is also called **typed first-order logic**, and the sorts called **types** (as in data type), but it is not the same as first-order type theory. Many-sorted first-order logic is often used in the study of second-order arithmetic.

When there are only finitely many sorts in a theory, many-sorted first-order logic can be reduced to single-sorted first-order logic. One introduces into the single-sorted theory a unary predicate symbol for each sort in the many-sorted theory, and adds an axiom saying that these unary predicates partition the domain of discourse. For example, if there are two sorts, one adds predicate symbols $P_1(x)$ and $P_2(x)$ and the axiom

$$\forall x(P_1(x) \vee P_2(x)) \wedge \neg\exists x(P_1(x) \wedge P_2(x))$$

Then the elements satisfying P_1 are thought of as elements of the first sort, and elements satisfying P_2 as elements of the second sort. One can quantify over each sort by using the corresponding predicate symbol to limit the range of quantification. For example, to say there is an element of the first sort satisfying formula $\varphi(x)$, one writes

$$\exists x(P_1(x) \wedge \phi(x))$$

1.8.3 Additional quantifiers

Additional quantifiers can be added to first-order logic.

- Sometimes it is useful to say that "*P(x)* holds for exactly one *x*", which can be expressed as $\exists!\ x\ P(x)$. This notation, called uniqueness quantification, may be taken to abbreviate a formula such as $\exists\ x\ (P(x) \wedge \forall\ y\ (P(y) \rightarrow (x = y)))$.
- **First-order logic with extra quantifiers** has new quantifiers *Qx*,..., with meanings such as "there are many *x* such that ...". Also see branching quantifiers and the plural quantifiers of George Boolos and others.
- **Bounded quantifiers** are often used in the study of set theory or arithmetic.

1.8.4 Infinitary logics

Main article: Infinitary logic

Infinitary logic allows infinitely long sentences. For example, one may allow a conjunction or disjunction of infinitely many formulas, or quantification over infinitely many variables. Infinitely long sentences arise in areas of mathematics including topology and model theory.

Infinitary logic generalizes first-order logic to allow formulas of infinite length. The most common way in which formulas can become infinite is through infinite conjunctions and disjunctions. However, it is also possible to admit generalized signatures in which function and relation symbols are allowed to have infinite arities, or in which quantifiers can bind infinitely many variables. Because an infinite formula cannot be represented by a finite string, it is necessary to choose some other representation of formulas; the usual representation in this context is a tree. Thus formulas are, essentially, identified with their parse trees, rather than with the strings being parsed.

The most commonly studied infinitary logics are denoted $L\alpha_\beta$, where α and β are each either cardinal numbers or the symbol ∞. In this notation, ordinary first-order logic is $L\omega\omega$. In the logic $L\infty\omega$, arbitrary conjunctions or disjunctions are allowed when building formulas, and there is an unlimited supply of variables. More generally, the logic that permits conjunctions or disjunctions with less than κ constituents is known as $L\kappa\omega$. For example, $L\omega_1\omega$ permits countable conjunctions and disjunctions.

The set of free variables in a formula of $L\kappa\omega$ can have any cardinality strictly less than κ, yet only finitely many of them can be in the scope of any quantifier when a formula appears as a subformula of another.[9] In other infinitary logics, a subformula may be in the scope of infinitely many quantifiers. For example, in $L\kappa\infty$, a single universal or existential quantifier may bind arbitrarily many variables simultaneously. Similarly, the logic $L\kappa\lambda$ permits simultaneous quantification over fewer than λ variables, as well as conjunctions and disjunctions of size less than κ.

1.8.5 Non-classical and modal logics

- **Intuitionistic first-order logic** uses intuitionistic rather than classical propositional calculus; for example, $\neg\neg\varphi$ need not be equivalent to φ.

- First-order **modal logic** allows one to describe other possible worlds as well as this contingently true world which we inhabit. In some versions, the set of possible worlds varies depending on which possible world one inhabits. Modal logic has extra *modal operators* with meanings which can be characterized informally as, for example "it is necessary that φ" (true in all possible worlds) and "it is possible that φ" (true in some possible world). With standard first-order logic we have a single domain and each predicate is assigned one extension. With first-order modal logic we have a *domain function* that assigns each possible world its own domain, so that each predicate gets an extension only relative to these possible worlds. This allows us to model cases where, for example, Alex is a Philosopher, but might have been a Mathematician, and might not have existed at all. In the first possible world $P(a)$ is true, in the second $P(a)$ is false, and in the third possible world there is no a in the domain at all.
- **first-order fuzzy logics** are first-order extensions of propositional fuzzy logics rather than classical propositional calculus.

1.8.6 Fixpoint logic

Fixpoint logic extends first-order logic by adding the closure under the least fixed points of positive operators.[10]

1.8.7 Higher-order logics

Main article: Higher-order logic

The characteristic feature of first-order logic is that individuals can be quantified, but not predicates. Thus

$$\exists a(\text{Phil}(a))$$

is a legal first-order formula, but

$$\exists \text{Phil}(\text{Phil}(a))$$

is not, in most formalizations of first-order logic. Second-order logic extends first-order logic by adding the latter type of quantification. Other higher-order logics allow quantification over even higher types than second-order logic permits. These higher types include relations between relations, functions from relations to relations between relations, and other higher-type objects. Thus the "first" in first-order logic describes the type of objects that can be quantified.

Unlike first-order logic, for which only one semantics is studied, there are several possible semantics for second-order logic. The most commonly employed semantics for second-order and higher-order logic is known as **full semantics**. The combination of additional quantifiers and the full semantics for these quantifiers makes higher-order logic stronger than first-order logic. In particular, the (semantic) logical consequence relation for second-order and higher-order logic is not semidecidable; there is no effective deduction system for second-order logic that is sound and complete under full semantics.

Second-order logic with full semantics is more expressive than first-order logic. For example, it is possible to create axiom systems in second-order logic that uniquely characterize the natural numbers and the real line. The cost of this expressiveness is that second-order and higher-order logics have fewer attractive metalogical properties than first-order logic. For example, the Löwenheim–Skolem theorem and compactness theorem of first-order logic become false when generalized to higher-order logics with full semantics.

1.9 Automated theorem proving and formal methods

Further information: First-order theorem proving

Automated theorem proving refers to the development of computer programs that search and find derivations (formal proofs) of mathematical theorems. Finding derivations is a difficult task because the search space can be very large; an

exhaustive search of every possible derivation is theoretically possible but computationally infeasible for many systems of interest in mathematics. Thus complicated heuristic functions are developed to attempt to find a derivation in less time than a blind search.

The related area of automated proof verification uses computer programs to check that human-created proofs are correct. Unlike complicated automated theorem provers, verification systems may be small enough that their correctness can be checked both by hand and through automated software verification. This validation of the proof verifier is needed to give confidence that any derivation labeled as "correct" is actually correct.

Some proof verifiers, such as Metamath, insist on having a complete derivation as input. Others, such as Mizar and Isabelle, take a well-formatted proof sketch (which may still be very long and detailed) and fill in the missing pieces by doing simple proof searches or applying known decision procedures: the resulting derivation is then verified by a small, core "kernel". Many such systems are primarily intended for interactive use by human mathematicians: these are known as proof assistants. They may also use formal logics that are stronger than first-order logic, such as type theory. Because a full derivation of any nontrivial result in a first-order deductive system will be extremely long for a human to write,[11] results are often formalized as a series of lemmas, for which derivations can be constructed separately.

Automated theorem provers are also used to implement formal verification in computer science. In this setting, theorem provers are used to verify the correctness of programs and of hardware such as processors with respect to a formal specification. Because such analysis is time-consuming and thus expensive, it is usually reserved for projects in which a malfunction would have grave human or financial consequences.

1.10 See also

- ACL2 — A Computational Logic for Applicative Common Lisp.
- Equiconsistency
- Extension by definitions
- Herbrandization
- Higher-order logic
- List of logic symbols
- Löwenheim number
- Prenex normal form
- Relational algebra
- Relational model
- Second-order logic
- Skolem normal form
- Tarski's World
- Truth table
- Type (model theory)
- Prolog

1.11 Notes

[1] First Order Logic CSC 2501: Symbolic Computation and AI, Dr. J.P.E. Hodgson, Saint Joseph's University, Philadelphia

[2] Mendelson, Elliott (1964). *Introduction to Mathematical Logic*. Van Nostrand Reinhold. p. 56.

[3] The word *language* is sometimes used as a synonym for signature, but this can be confusing because "language" can also refer to the set of formulas.

[4] More precisely, there is only one language of each variant of one-sorted first-order logic: with or without equality, with or without functions, with or without propositional variables, ...

[5] Some authors who use the term "well-formed formula" use "formula" to mean any string of symbols from the alphabet. However, most authors in mathematical logic use "formula" to mean "well-formed formula" and have no term for non-well-formed formulas. In every context, it is only the well-formed formulas that are of interest.

[6] The SMT-LIB Standard: Version 2.0, by Clark Barrett, Aaron Stump, and Cesare Tinelli. http://smtlib.cs.uiowa.edu/language.shtml

[7] e.g. the matrix shown at 4

[8] e.g. the matrix shown at 2

[9] Some authors only admit formulas with finitely many free variables in $L\kappa\omega$, and more generally only formulas with $< \lambda$ free variables in $L\kappa\lambda$.

[10] Bosse, Uwe (1993). "An Ehrenfeucht–Fraïssé game for fixpoint logic and stratified fixpoint logic". In Börger, Egon. *Computer Science Logic: 6th Workshop, CSL'92, San Miniato, Italy, September 28 - October 2, 1992. Selected Papers*. Lecture Notes in Computer Science **702**. Springer-Verlag. pp. 100–114. ISBN 3-540-56992-8. Zbl 0808.03024.

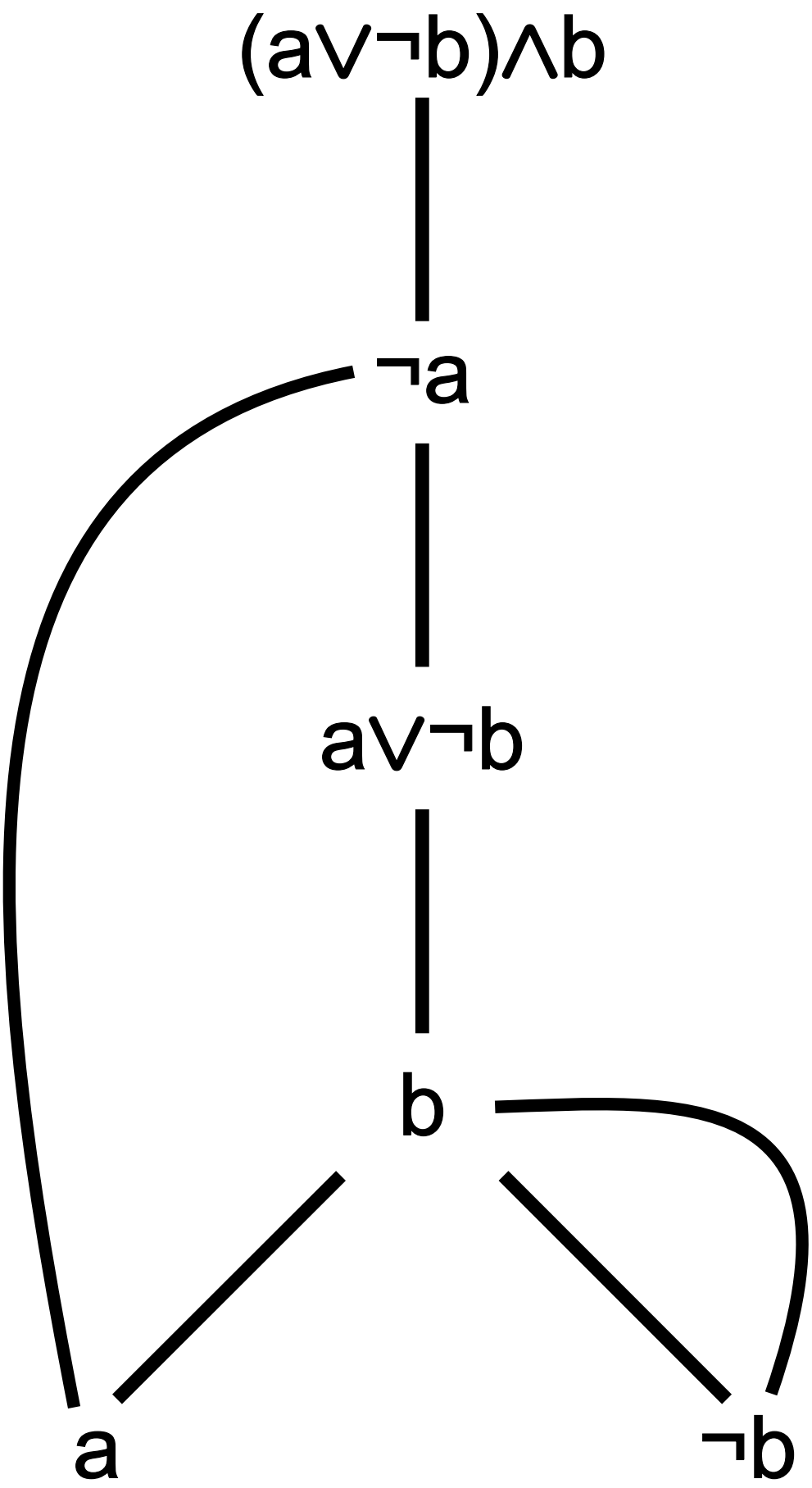
[11] Avigad *et al.* (2007) discuss the process of formally verifying a proof of the prime number theorem. The formalized proof required approximately 30,000 lines of input to the Isabelle proof verifier.

1.12 References

- Andrews, Peter B. (2002); *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*, 2nd ed., Berlin: Kluwer Academic Publishers. Available from Springer.
- Avigad, Jeremy; Donnelly, Kevin; Gray, David; and Raff, Paul (2007); "A formally verified proof of the prime number theorem", *ACM Transactions on Computational Logic*, vol. 9 no. 1 doi:10.1145/1297658.1297660
- Barwise, Jon (1977); "An Introduction to First-Order Logic", in Barwise, Jon, ed. (1982). *Handbook of Mathematical Logic*. Studies in Logic and the Foundations of Mathematics. Amsterdam, NL: North-Holland. ISBN 978-0-444-86388-1.
- Barwise, Jon; and Etchemendy, John (2000); *Language Proof and Logic*, Stanford, CA: CSLI Publications (Distributed by the University of Chicago Press)
- Bocheński, Józef Maria (2007); *A Précis of Mathematical Logic*, Dordrecht, NL: D. Reidel, translated from the French and German editions by Otto Bird
- Ferreirós, José (2001); *The Road to Modern Logic — An Interpretation*, Bulletin of Symbolic Logic, Volume 7, Issue 4, 2001, pp. 441–484, DOI 10.2307/2687794, JStor
- Gamut, L. T. F. (1991); *Logic, Language, and Meaning, Volume 2: Intensional Logic and Logical Grammar*, Chicago, IL: University of Chicago Press, ISBN 0-226-28088-8
- Hilbert, David; and Ackermann, Wilhelm (1950); *Principles of Mathematical Logic*, Chelsea (English translation of *Grundzüge der theoretischen Logik*, 1928 German first edition)
- Hodges, Wilfrid (2001); "Classical Logic I: First Order Logic", in Goble, Lou (ed.); *The Blackwell Guide to Philosophical Logic*, Blackwell
- Ebbinghaus, Heinz-Dieter; Flum, Jörg; and Thomas, Wolfgang (1994); *Mathematical Logic*, Undergraduate Texts in Mathematics, Berlin, DE/New York, NY: Springer-Verlag, Second Edition, ISBN 978-0-387-94258-2
- Rautenberg, Wolfgang (2010), *A Concise Introduction to Mathematical Logic* (3rd ed.), New York, NY: Springer Science+Business Media, doi:10.1007/978-1-4419-1221-3, ISBN 978-1-4419-1220-6
- Tarski, Alfred and Givant, Steven (1987); *A Formalization of Set Theory without Variables*. Vol.41 of American Mathematical Society colloquium publications, Providence RI: American Mathematical Society, ISBN 978-0821810415.

1.13 External links

- Hazewinkel, Michiel, ed. (2001), "Predicate calculus", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Stanford Encyclopedia of Philosophy: Shapiro, Stewart; "Classical Logic". Covers syntax, model theory, and metatheory for first-order logic in the natural deduction style.
- Magnus, P. D.; *forall x: an introduction to formal logic*. Covers formal semantics and proof theory for first-order logic.
- Metamath: an ongoing online project to reconstruct mathematics as a huge first-order theory, using first-order logic and the axiomatic set theory ZFC. *Principia Mathematica* modernized.
- Podnieks, Karl; *Introduction to mathematical logic*
- *Cambridge Mathematics Tripos Notes* (typeset by John Fremlin). These notes cover part of a past Cambridge Mathematics Tripos course taught to undergraduates students (usually) within their third year. The course is entitled "Logic, Computation and Set Theory" and covers Ordinals and cardinals, Posets and Zorn's Lemma, Propositional logic, Predicate logic, Set theory and Consistency issues related to ZFC and other set theories.
- Tree Proof Generator can validate or invalidate formulas of first-order logic through the semantic tableaux method.


(a∨¬b)∧b
¬a
a∨¬b
b
a
¬b

Chapter 2

Peano axioms

In mathematical logic, the **Peano axioms**, also known as the **Dedekind–Peano axioms** or the **Peano postulates**, are a set of axioms for the natural numbers presented by the 19th century Italian mathematician Giuseppe Peano. These axioms have been used nearly unchanged in a number of metamathematical investigations, including research into fundamental questions of whether number theory is consistent and complete.

The need to formalize arithmetic was not well appreciated until the work of Hermann Grassmann, who showed in the 1860s that many facts in arithmetic could be derived from more basic facts about the successor operation and induction.[1] In 1881, Charles Sanders Peirce provided an axiomatization of natural-number arithmetic.[2] In 1888, Richard Dedekind proposed another axiomatization of natural-number arithmetic, and in 1889, Peano published a more precisely formulated version of them as a collection of axioms in his book, *The principles of arithmetic presented by a new method* (Latin: *Arithmetices principia, nova methodo exposita*).

The Peano axioms contain three types of statements. The first axiom asserts the existence of at least one member of the set of natural numbers. The next four are general statements about equality; in modern treatments these are often not taken as part of the Peano axioms, but rather as axioms of the "underlying logic".[3] The next three axioms are first-order statements about natural numbers expressing the fundamental properties of the successor operation. The ninth, final axiom is a second order statement of the principle of mathematical induction over the natural numbers. A weaker first-order system called **Peano arithmetic** is obtained by explicitly adding the addition and multiplication operation symbols and replacing the second-order induction axiom with a first-order axiom schema.

2.1 Formulation

When Peano formulated his axioms, the language of mathematical logic was in its infancy. The system of logical notation he created to present the axioms did not prove to be popular, although it was the genesis of the modern notation for set membership ($\in$, which comes from Peano's ε) and implication ($\supset$, which comes from Peano's reversed 'C'.) Peano maintained a clear distinction between mathematical and logical symbols, which was not yet common in mathematics; such a separation had first been introduced in the *Begriffsschrift* by Gottlob Frege, published in 1879.[4] Peano was unaware of Frege's work and independently recreated his logical apparatus based on the work of Boole and Schröder.[5]

The Peano axioms define the arithmetical properties of *natural numbers*, usually represented as a set $\mathbf{N}$ or $\mathbb{N}$. The signature (a formal language's non-logical symbols) for the axioms includes a constant symbol 0 and a unary function symbol S.

The constant 0 is assumed to be a natural number:

1. 0 is a natural number.

The next four axioms describe the equality relation. Since they are logically valid in first-order logic with equality, they are not considered to be part of "the Peano axioms" in modern treatments.[6]

1. For every natural number x, $x = x$. That is, equality is reflexive.

2. For all natural numbers x and y, if $x = y$, then $y = x$. That is, equality is symmetric.

3. For all natural numbers x, y and z, if $x = y$ and $y = z$, then $x = z$. That is, equality is transitive.

4. For all a and b, if b is a natural number and $a = b$, then a is also a natural number. That is, the natural numbers are closed under equality.

The remaining axioms define the arithmetical properties of the natural numbers. The naturals are assumed to be closed under a single-valued "successor" function S.

1. For every natural number n, $S(n)$ is a natural number.

2. For all natural numbers m and n, $m = n$ if and only if $S(m) = S(n)$. That is, S is an injection.

3. For every natural number n, $S(n) = 0$ is false. That is, there is no natural number whose successor is 0.

Peano's original formulation of the axioms used 1 instead of 0 as the "first" natural number.[7] This choice is arbitrary, as axiom 1 does not endow the constant 0 with any additional properties. However, because 0 is the additive identity in arithmetic, most modern formulations of the Peano axioms start from 0. Axioms 1, 6, 7, 8 define a unary representation of the intuitive notion of natural numbers: the number 1 can be defined as $S(0)$, 2 as $S(S(0))$, etc. However, considering the notion of natural numbers as can be derived from the axioms, axioms 1, 6, 7, 8 do not imply that the successor function generates all the natural numbers different than 0. Put differently, they do not guarantee that every natural number other than zero must succeed some other natural number.

The intuitive notion that all natural numbers observe a succession relation with one (in the case of the starting number) or two (for all other numbers) other numbers requires an additional axiom, which is sometimes called the *axiom of induction*.

1. If K is a set such that:

   - 0 is in K, and
   - for every natural number n, if n is in K, then $S(n)$ is in K,

   then K contains every natural number.

The induction axiom is sometimes stated in the following form:

1. If φ is a unary predicate such that:

   - $\varphi(0)$ is true, and
   - for every natural number n, if $\varphi(n)$ is true, then $\varphi(S(n))$ is true,

   then $\varphi(n)$ is true for every natural number n.

In Peano's original formulation, the induction axiom is a second-order axiom. It is now common to replace this second-order principle with a weaker first-order induction scheme. There are important differences between the second-order and first-order formulations, as discussed in the section § Models below.

2.2 Arithmetic

The Peano axioms can be augmented with the operations of addition and multiplication and the usual total (linear) ordering on **N**. The respective functions and relations are constructed in second-order logic, and are shown to be unique using the Peano axioms.

2.2.1 Addition

Addition is a function that maps two natural numbers (two elements of **N**) to another one. It is defined recursively as:

$$\begin{aligned} a + 0 &= a, && (1)\\ a + S(b) &= S(a + b). && (2) \end{aligned}$$

For example:

$$\begin{aligned} a + 1 &= a + S(0) && \text{by definition}\\ &= S(a + 0) && \text{using (2)}\\ &= S(a), && \text{using (1)} \end{aligned}$$

$$\begin{aligned} a + 2 &= a + S(1) && \text{by definition}\\ &= S(a + 1) && \text{using (2)}\\ &= S(S(a)) && \text{using } a + 1 = S(a) \end{aligned}$$

$$\begin{aligned} a + 3 &= a + S(2) && \text{by definition}\\ &= S(a + 2) && \text{using (2)}\\ &= S(S(S(a)) && \text{using } a + 2 = S(S(a)) \end{aligned}$$

etc.

The structure (**N**, +) is a commutative semigroup with identity element 0. (**N**, +) is also a cancellative magma, and thus embeddable in a group. The smallest group embedding **N** is the integers.

2.2.2 Multiplication

Similarly, multiplication is a function mapping two natural numbers to another one. Given addition, it is defined recursively as:

$$\begin{aligned} a \cdot 0 &= 0,\\ a \cdot S(b) &= a + (a \cdot b). \end{aligned}$$

It is easy to see that $S(0)$ (or "1", in the familiar language of decimal representation) is the multiplicative identity:

$$a \cdot S(0) = a + (a \cdot 0) = a + 0 = a$$

Moreover, multiplication distributes over addition:

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c).$$

Thus, (**N**, +, 0, ·, $S(0)$) is a commutative semiring.

2.2.3 Inequalities

The usual total order relation ≤ on natural numbers can be defined as follows, assuming 0 is a natural number:

For all $a, b \in \mathbf{N}$, $a \leq b$ if and only if there exists some $c \in \mathbf{N}$ such that $a + c = b$.

This relation is stable under addition and multiplication: for $a, b, c \in \mathbf{N}$, if $a \leq b$, then:

- $a + c \leq b + c$, and
- $a \cdot c \leq b \cdot c$.

Thus, the structure $(\mathbf{N}, +, \cdot, 1, 0, \leq)$ is an ordered semiring; because there is no natural number between 0 and 1, it is a discrete ordered semiring.

The axiom of induction is sometimes stated in the following *strong* form, making use of the $\leq$ order:

> For any predicate φ, if
>
> - $\varphi(0)$ is true, and
> - for every $n, k \in \mathbf{N}$, if $k \leq n$ implies that $\varphi(k)$ is true, then $\varphi(S(n))$ is true,
>
> then for every $n \in \mathbf{N}$, $\varphi(n)$ is true.

This form of the induction axiom is a simple consequence of the standard formulation, but is often better suited for reasoning about the $\leq$ order. For example, to show that the naturals are well-ordered—every nonempty subset of $\mathbf{N}$ has a least element—one can reason as follows. Let a nonempty $X \subseteq \mathbf{N}$ be given and assume X has no least element.

- Because 0 is the least element of $\mathbf{N}$, it must be that $0 \notin X$.
- For any $n \in \mathbf{N}$, suppose for every $k \leq n$, $k \notin X$. Then $S(n) \notin X$, for otherwise it would be the least element of X.

Thus, by the strong induction principle, for every $n \in \mathbf{N}$, $n \notin X$. Thus, $X \cap \mathbf{N} = \varnothing$, which contradicts X being a nonempty subset of $\mathbf{N}$. Thus X has a least element.

2.3 First-order theory of arithmetic

All of the Peano axioms except the ninth axiom (the induction axiom) are statements in first-order logic.[8] The arithmetical operations of addition and multiplication and the order relation can also be defined using first-order axioms. The axiom of induction is in second-order, since it quantifies over predicates (equivalently, sets of natural numbers rather than natural numbers), but it can be transformed into a first-order *axiom schema* of induction. Such a schema includes one axiom per predicate definable in the first-order language of Peano arithmetic, making it weaker than the second-order axiom.[9]

First-order axiomatizations of Peano arithmetic have an important limitation, however. In second-order logic, it is possible to define the addition and multiplication operations from the successor operation, but this cannot be done in the more restrictive setting of first-order logic. Therefore, the addition and multiplication operations are directly included in the signature of Peano arithmetic, and axioms are included that relate the three operations to each other.

The following list of axioms (along with the usual axioms of equality), which contains six of the seven axioms of Robinson arithmetic, is sufficient for this purpose:[10]

- $\forall x \in \mathbf{N}.\ 0 \neq S(x)$
- $\forall x, y \in \mathbf{N}.\ S(x) = S(y) \Rightarrow x = y$
- $\forall x \in \mathbf{N}.\ x + 0 = x$
- $\forall x, y \in \mathbf{N}.\ x + S(y) = S(x + y)$
- $\forall x \in \mathbf{N}.\ x \cdot 0 = 0$
- $\forall x, y \in \mathbf{N}.\ x \cdot S(y) = x \cdot y + x$

In addition to this list of numerical axioms, Peano arithmetic contains the induction schema, which consists of a countably infinite set of axioms. For each formula $\varphi(x, y_1, \ldots, yk)$ in the language of Peano arithmetic, the **first-order induction axiom** for φ is the sentence

$$\forall \bar{y}(\varphi(0, \bar{y}) \wedge \forall x(\varphi(x, \bar{y}) \Rightarrow \varphi(S(x), \bar{y})) \Rightarrow \forall x \varphi(x, \bar{y}))$$

where $\bar{y}$ is an abbreviation for $y_1,\ldots,yk$. The first-order induction schema includes every instance of the first-order induction axiom, that is, it includes the induction axiom for every formula φ.

2.3.1 Equivalent axiomatizations

There are many different, but equivalent, axiomatizations of Peano arithmetic. While some axiomatizations, such as the one just described, use a signature that only has symbols for 0 and the successor, addition, and multiplications operations, other axiomatizations use the language of ordered semirings, including an additional order relation symbol. One such axiomatization begins with the following axioms that describe a discrete ordered semiring.[11]

1. $\forall x, y, z \in \mathbf{N} \quad (x + y) + z = x + (y + z)$, i.e., addition is associative.
2. $\forall x, y \in \mathbf{N} \quad x + y = y + x$, i.e., addition is commutative.
3. $\forall x, y, z \in \mathbf{N} \quad (x \cdot y) \cdot z = x \cdot (y \cdot z)$, i.e., multiplication is associative.
4. $\forall x, y \in \mathbf{N} \quad x \cdot y = y \cdot x$, i.e., multiplication is commutative.
5. $\forall x, y, z \in \mathbf{N} \quad x \cdot (y + z) = (x \cdot y) + (x \cdot z)$, i.e., the distributive law.
6. $\forall x \in \mathbf{N} \quad x + 0 = x \wedge x \cdot 0 = 0$, i.e., zero is the identity element for addition.
7. $\forall x \in \mathbf{N} \quad x \cdot 1 = x$, i.e., one is the identity element for multiplication.
8. $\forall x, y, z \in \mathbf{N} \quad x < y \wedge y < z \Rightarrow x < z$, i.e., the '<' operator is transitive.
9. $\forall x \in \mathbf{N} \quad \neg(x < x)$, i.e., the '<' operator is irreflexive.
10. $\forall x, y \in \mathbf{N} \quad x < y \vee x = y \vee y < x$.
11. $\forall x, y, z \in \mathbf{N} \quad x < y \Rightarrow x + z < y + z$.
12. $\forall x, y, z \in \mathbf{N} \quad 0 < z \wedge x < y \Rightarrow x \cdot z < y \cdot z$.
13. $\forall x, y \in \mathbf{N} \quad x < y \Rightarrow \exists z \in N x + z = y$.
14. $0 < 1 \wedge \forall x \in \mathbf{N} \quad x > 0 \Rightarrow x \geq 1$.
15. $\forall x \in \mathbf{N} \,.\, x \geq 0$.

The theory defined by these axioms is known as $\mathbf{PA}^-$; PA is obtained by adding the first-order induction schema.

An important property of PA^- is that any structure M satisfying this theory has an initial segment (ordered by $\leq$) isomorphic to $\mathbf{N}$. Elements of $M \setminus \mathbf{N}$ are known as **nonstandard** elements.

2.4 Models

A model of the Peano axioms is a triple ($\mathbf{N}$, 0, S), where $\mathbf{N}$ is a (necessarily infinite) set, $0 \in \mathbf{N}$ and $S : \mathbf{N} \to \mathbf{N}$ satisfies the axioms above. Dedekind proved in his 1888 book, *What are numbers and what should they be* (German: *Was sind und was sollen die Zahlen*) that any two models of the Peano axioms (including the second-order induction axiom) are isomorphic. In particular, given two models ($\mathbf{N}A$, $0A$, SA) and ($\mathbf{N}B$, $0B$, SB) of the Peano axioms, there is a unique homomorphism $f : \mathbf{N}A \to \mathbf{N}B$ satisfying

$$\begin{aligned} f(0_A) &= 0_B \\ f(S_A(n)) &= S_B(f(n)) \end{aligned}$$

and it is a bijection. This means that second-order Peano axioms are categorical. This is not the case with any first-order reformulation of the Peano axioms, however.

2.4.1 Nonstandard models

Although the usual natural numbers satisfy the axioms of PA, there are other **non-standard models** as well; the compactness theorem implies that the existence of nonstandard elements cannot be excluded in first-order logic. The upward Löwenheim–Skolem theorem shows that there are nonstandard models of PA of all infinite cardinalities. This is not the case for the original (second-order) Peano axioms, which have only one model, up to isomorphism. This illustrates one way the first-order system PA is weaker than the second-order Peano axioms.

When interpreted as a proof within a first-order set theory, such as ZFC, Dedekind's categoricity proof for PA shows that each model of set theory has a unique model of the Peano axioms, up to isomorphism, that embeds as an initial segment of all other models of PA contained within that model of set theory. In the standard model of set theory, this smallest model of PA is the standard model of PA; however, in a nonstandard model of set theory, it may be a nonstandard model of PA. This situation cannot be avoided with any first-order formalization of set theory.

It is natural to ask whether a countable nonstandard model can be explicitly constructed. The answer is affirmative as Skolem in 1933 provided an explicit construction of such a nonstandard model. On the other hand, Tennenbaum's theorem, proved in 1959, shows that there is no countable nonstandard model of PA in which either the addition or multiplication operation is computable.[12] This result shows it is difficult to be completely explicit in describing the addition and multiplication operations of a countable nonstandard model of PA. However, there is only one possible order type of a countable nonstandard model. Letting ω be the order type of the natural numbers, ζ be the order type of the integers, and η be the order type of the rationals, the order type of any countable nonstandard model of PA is $\omega + \zeta \cdot \eta$, which can be visualized as a copy of the natural numbers followed by a dense linear ordering of copies of the integers.

2.4.2 Set-theoretic models

Main article: Set-theoretic definition of natural numbers

The Peano axioms can be derived from set theoretic constructions of the natural numbers and axioms of set theory such as the ZF.[13] The standard construction of the naturals, due to John von Neumann, starts from a definition of 0 as the empty set, $\varnothing$, and an operator s on sets defined as:

$$s(a) = a \cup \{\ a\ \}.$$

The set of natural numbers $\mathbf{N}$ is defined as the intersection of all sets closed under s that contain the empty set. Each natural number is equal (as a set) to the set of natural numbers less than it:

$$0 = \emptyset$$
$$1 = s(0) = s(\emptyset) = \emptyset \cup \{\emptyset\} = \{\emptyset\} = \{0\}$$
$$2 = s(1) = s(\{0\}) = \{0\} \cup \{\{0\}\} = \{0, \{0\}\} = \{0, 1\}$$
$$3 = s(2) = s(\{0, 1\}) = \{0, 1\} \cup \{\{0, 1\}\} = \{0, 1, \{0, 1\}\} = \{0, 1, 2\}$$

and so on. The set $\mathbf{N}$ together with 0 and the successor function $s : \mathbf{N} \to \mathbf{N}$ satisfies the Peano axioms.

Peano arithmetic is equiconsistent with several weak systems of set theory.[14] One such system is ZFC with the axiom of infinity replaced by its negation. Another such system consists of general set theory (extensionality, existence of the empty set, and the axiom of adjunction), augmented by an axiom schema stating that a property that holds for the empty set and holds of an adjunction whenever it holds of the adjunct must hold for all sets.

2.4.3 Interpretation in category theory

The Peano axioms can also be understood using category theory. Let C be a category with terminal object $1C$, and define the category of pointed unary systems, $\mathrm{US}_1(C)$ as follows:

- The objects of $\mathrm{US}_1(C)$ are triples $(X, 0X, SX)$ where X is an object of C, and $0X : 1C \to X$ and $SX : X \to X$ are C-morphisms.

- A morphism $\varphi : (X, 0X, SX) \to (Y, 0Y, SY)$ is a C-morphism $\varphi : X \to Y$ with $\varphi\, 0X = 0Y$ and $\varphi\, SX = SY\, \varphi$.

Then C is said to satisfy the Dedekind–Peano axioms if $\mathrm{US}_1(C)$ has an initial object; this initial object is known as a natural number object in C. If $(\mathbf{N}, 0, S)$ is this initial object, and $(X, 0X, SX)$ is any other object, then the unique map $u : (\mathbf{N}, 0, S) \to (X, 0X, SX)$ is such that

$$\begin{aligned} u0 &= 0_X, \\ u(Sx) &= S_X(ux). \end{aligned}$$

This is precisely the recursive definition of $0X$ and SX.

2.5 Consistency

Further information: Hilbert's second problem and Consistency

When the Peano axioms were first proposed, Bertrand Russell and others agreed that these axioms implicitly defined what we mean by a "natural number".[15] Henri Poincaré was more cautious, saying they only defined natural numbers if they were *consistent*; if there is a proof that starts from just these axioms and derives a contradiction such as 0 = 1, then the axioms are inconsistent, and don't define anything..[16] In 1900, David Hilbert posed the problem of proving their consistency using only finitistic methods as the second of his twenty-three problems.[17] In 1931, Kurt Gödel proved his second incompleteness theorem, which shows that such a consistency proof cannot be formalized within Peano arithmetic itself.[18]

Although it is widely claimed that Gödel's theorem rules out the possibility of a finitistic consistency proof for Peano arithmetic, this depends on exactly what one means by a finitistic proof. Gödel himself pointed out the possibility of giving a finitistic consistency proof of Peano arithmetic or stronger systems by using finitistic methods that are not formalizable in Peano arithmetic, and in 1958, Gödel published a method for proving the consistency of arithmetic using type theory.[19] In 1936, Gerhard Gentzen gave a proof of the consistency of Peano's axioms, using transfinite induction up to an ordinal called ε_0.[20] Gentzen explained: "The aim of the present paper is to prove the consistency of elementary number theory or, rather, to reduce the question of consistency to certain fundamental principles". Gentzen's proof is arguably finitistic, since the transfinite ordinal ε_0 can be encoded in terms of finite objects (for example, as a Turing machine describing a suitable order on the integers, or more abstractly as consisting of the finite trees, suitably linearly ordered). Whether or not Gentzen's proof meets the requirements Hilbert envisioned is unclear: there is no generally accepted definition of exactly what is meant by a finitistic proof, and Hilbert himself never gave a precise definition.

The vast majority of contemporary mathematicians believe that Peano's axioms are consistent, relying either on intuition or the acceptance of a consistency proof such as Gentzen's proof. The small number of mathematicians who advocate ultrafinitism reject Peano's axioms because the axioms require an infinite set of natural numbers.

2.6 See also

- Foundations of mathematics
- Frege's theorem
- Goodstein's theorem
- Non-standard model of arithmetic
- Paris–Harrington theorem
- Presburger arithmetic
- Robinson arithmetic
- Second-order arithmetic

2.7 Footnotes

[1] Grassmann 1861

[2] Peirce 1881, Shields 1997

[3] van Heijenoort 1967, p. 94

[4] van Heijenoort 1967, p. 2

[5] van Heijenoort 1967, p. 83

[6] van Heijenoort 1967, p. 83

[7] Peano 1889, p. 1

[8] Partee *et al.* 2012, p. 215

[9] Harsanyi 1983

[10] Mendelson 1997, p. 155

[11] Kaye 1991, pp. 16–18

[12] Kaye 1991, Section 11.3

[13] Suppes 1960, Hatcher 1982

[14] Tarski & Givant 1987, Section 7.6

[15] Fritz 1952, p. 137
An illustration of 'interpretation' is Russell's own definition of 'cardinal number'. The uninterpreted system in this case is Peano's axioms for the number system, whose three primitive ideas and five axioms, Peano believed, were sufficient to enable one to derive all the properties of the system of natural numbers. Actually, Russell maintains, Peano's axioms define any progression of the form $x_0, x_1, x_2, \ldots, x_n, \ldots$ of which the series of the natural numbers is one instance.

[16] Gray 2013, p. 133
So Poincaré turned to see whether logicism could generate arithmetic, more precisely, the arithmetic of ordinals. Couturat, said Poincaré, had accepted the Peano axioms as a definition of a number. But this will not do. The axioms cannot be shown to be free of contradiction by finding examples of them, and any attempt to show that they were contradiction-free by examining the totality of their implications would require the very principle of mathematical induction Couturat believed they implied. For (in a further passage dropped from S&M) either one assumed the principle in order to prove it, which would only prove that if it is true it is not self-contradictory, which says nothing; or one used the principle in another form than the one stated, in which case one must show that the number of steps in one's reasoning was an integer according to the new definition, but this could not be done (1905c, 834).

[17] Hilbert 1900

[18] Gödel 1931

[19] Gödel 1958

[20] Gentzen 1936

2.8 References

- Davis, Martin (1974). *Computability. Notes by Barry Jacobs*. Courant Institute of Mathematical Sciences, New York University.
- Dedekind, Richard (1888). *Was sind und was sollen die Zahlen?* [*What are and what should the numbers be?*] (PDF). Vieweg. Retrieved 4 July 2016..
  - Two English translations:
    - Beman, Wooster, Woodruff (1901). *Essays on the Theory of Numbers* (PDF). Dover.
    - Ewald, William B. *From Kant to Hilbert: A Source Book in the Foundations of Mathematics, 2 vols*. Oxford University Press. pp. 787–832.

- Fritz Jr, Charles A. (1952). *Bertrand Russell's construction of the external world*.
- Gentzen, Gerhard (1936). Reprinted in English translation in his 1969 *Collected works*, M. E. Szabo, ed.. "Die Widerspruchsfreiheit der reinen Zahlentheorie". *Mathematische Annalen* (Amsterdam: North-Holland) **112**: 132–213.
- Gödel, Kurt (1931). See On Formally Undecidable Propositions of Principia Mathematica and Related Systems for details on English translations.. "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I" (PDF). *Monatshefte für Mathematik* **38**: 173–198. doi:10.1007/bf01700692.
- Gödel, Kurt (1958). Reprinted in English translation in 1990. Gödel's *Collected Works*, Vol II. Solomon Feferman et al., eds.. "Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes". *Dialectica* (Oxford University Press) **12**: 280–287.
- Grassmann, Hermann (1861). "Lehrbuch der Arithmetik" [A tutorial in arithmetic] (PDF). Enslin.
- Gray, Jeremy (2013). *Henri Poincaré: A scientific biography*. Princeton University Press. p. 133.
- Harsanyi, John C. (1983). "Mathematics, the empirical facts and logical necessity". *Erkenntniss* **19**: 167 ff.
- Hatcher, William S. (1982). *The Logical Foundations of Mathematics*. Derives the Peano axioms (called **S**) from several axiomatic set theories and from category theory. Pergamon.
- Hilbert, David (1902). English translation by Maby Winton, 1902. "Mathematische Probleme" [Mathematical Problems]. *Bulletin of the American Mathematical Society* **8**: 437–479.
- Kaye, Richard (1991). *Models of Peano arithmetic*. Oxford University Press. ISBN 0-19-853213-X.
- Landau, Edmund (1965). *Grundlagen Der Analysis*. Derives the basic number systems from the Peano axioms. English/German vocabulary included. AMS Chelsea Publishing. ISBN 978-0-8284-0141-8.
- Mendelson, Elliott (1997). *Introduction to Mathematical Logic* (4 ed.). Chapman & Hall.
- Partee, Barbara; Ter Meulen, Alice; Wall, Robert (2012). *Mathematical Methods in Linguistics*. Springer.
- Peirce, C. S. (1881). "On the Logic of Number". *American Journal of Mathematics* **4**: 85–95. doi:10.2307/2369151. JSTOR 2369151. MR 1507856.
- Shields, Paul (1997). "3. Peirce's Axiomatization of Arithmetic". In Houser, Nathan; Roberts, Don D.; Van Evra, James. *Studies in the Logic of Charles Sanders Peirce*.
- Suppes, Patrick (1960). *Axiomatic Set Theory*. Derives the Peano axioms from ZFC. Dover. ISBN 0-486-61630-4.
- Tarski, Alfred; Givant, Steven (1987). *A Formalization of Set Theory without Variables*. AMS Colloquium Publications, vol. 41. American Mathematical Society. ISBN 978-0-8218-1041-5.
- van Heijenoort, Jean (1967). *From Frege to Godel: A Source Book in Mathematical Logic, 1879–1931*. Cambridge, Mass: Harvard University Press. ISBN 9780674324497.
  - Contains translations of the following two papers, with valuable commentary:
    - Dedekind, Richard (1890). *Letter to Keferstein*. On p. 100, he restates and defends his axioms of 1888. pp. 98–103.
    - Peano, Giuseppe (1889). *Arithmetices principia, nova methodo exposita* [*The principles of arithmetic, presented by a new method*]. An excerpt of the treatise where Peano first presented his axioms, and recursively defined arithmetical operations. pp. 83–97.

2.9 Further reading

- Raymond M. Smullyan (19 September 2013). *The Godelian Puzzle Book: Puzzles, Paradoxes and Proofs*. Courier Corporation. ISBN 978-0-486-49705-1.

2.10 External links

- Internet Encyclopedia of Philosophy: "Henri Poincaré" by Mauro Murzi. Includes a discussion of Poincaré's critique of the Peano's axioms.
- First-order arithmetic, a chapter of a book on the incompleteness theorems by Karl Podnieks.
- Hazewinkel, Michiel, ed. (2001), "Peano axioms", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Peano arithmetic at PlanetMath.org.
- Weisstein, Eric W., "Peano's Axioms", *MathWorld*.
- What are numbers, and what is their meaning?: Dedekind commentary on Dedekind's work, Stanley N. Burris, 2001.

Chapter 3

Zermelo–Fraenkel set theory

"ZFC" redirects here. For other uses, see ZFC (disambiguation).

In mathematics, **Zermelo–Fraenkel set theory**, named after mathematicians Ernst Zermelo and Abraham Fraenkel, is one of several axiomatic systems that were proposed in the early twentieth century to formulate a theory of sets free of paradoxes such as Russell's paradox. Zermelo–Fraenkel set theory with the historically controversial axiom of choice included is commonly abbreviated **ZFC**, where C stands for choice.[1] Many authors use **ZF** to refer to the axioms of Zermelo–Fraenkel set theory with the axiom of choice excluded. Today ZFC is the standard form of axiomatic set theory and as such is the most common foundation of mathematics.

ZFC is intended to formalize a single primitive notion, that of a hereditary well-founded set, so that all entities in the universe of discourse are such sets. Thus the axioms of ZFC refer only to pure sets and prevent its models from containing urelements (elements of sets that are not themselves sets). Furthermore, proper classes (collections of mathematical objects defined by a property shared by their members which are too big to be sets) can only be treated indirectly. Specifically, ZFC does not allow for the existence of a universal set (a set containing all sets) nor for unrestricted comprehension, thereby avoiding Russell's paradox. Von Neumann–Bernays–Gödel set theory (NBG) is a commonly used conservative extension of ZFC that does allow explicit treatment of proper classes.

Formally, ZFC is a one-sorted theory in first-order logic. The signature has equality and a single primitive binary relation, set membership, which is usually denoted $\in$. The formula $a \in b$ means that the set a is a member of the set b (which is also read, "a is an element of b" or "a is in b").

There are many equivalent formulations of the ZFC axioms. Most of the ZFC axioms state the existence of particular sets defined from other sets. For example, the axiom of pairing says that given any two sets a and b there is a new set $\{a, b\}$ containing exactly a and b. Other axioms describe properties of set membership. A goal of the ZFC axioms is that each axiom should be true if interpreted as a statement about the collection of all sets in the von Neumann universe (also known as the cumulative hierarchy).

The metamathematics of ZFC has been extensively studied. Landmark results in this area established the independence of the continuum hypothesis from ZFC, and of the axiom of choice from the remaining ZFC axioms. The consistency of a theory such as ZFC cannot be proved within the theory itself.

3.1 History

In 1908, Ernst Zermelo proposed the first axiomatic set theory, Zermelo set theory. However, as first pointed out by Abraham Fraenkel in a 1921 letter to Zermelo, this theory was incapable of proving the existence of certain sets and cardinal numbers whose existence was taken for granted by most set theorists of the time, notably the cardinal number $\aleph\omega$ and the set $\{\mathrm{Z}_0, \wp(\mathrm{Z}_0), \wp(\wp(\mathrm{Z}_0)),...\}$, where Z_0 is any infinite set and $\wp$ is the power set operation.[2] Moreover, one of Zermelo's axioms invoked a concept, that of a "definite" property, whose operational meaning was not clear. In 1922, Fraenkel and Thoralf Skolem independently proposed operationalizing a "definite" property as one that could be formulated as a first order theory whose atomic formulas were limited to set membership and identity. They also independently proposed replacing the axiom schema of specification with the axiom schema of replacement. Appending this schema, as well as the axiom of regularity (first proposed by Dimitry Mirimanoff in 1917), to Zermelo set theory yields the theory denoted by **ZF**. Adding to ZF either the axiom of choice (AC) or a

statement that is equivalent to it yields ZFC.

3.2 Axioms

There are many equivalent formulations of the ZFC axioms; for a discussion of this see Fraenkel, Bar-Hillel & Lévy 1973. The following particular axiom set is from Kunen (1980). The axioms per se are expressed in the symbolism of first order logic. The associated English prose is only intended to aid the intuition.

All formulations of ZFC imply that at least one set exists. Kunen includes an axiom that directly asserts the existence of a set, in addition to the axioms given below (although he notes that he does so only "for emphasis").[3] Its omission here can be justified in two ways. First, in the standard semantics of first-order logic in which ZFC is typically formalized, the domain of discourse must be nonempty. Hence, it is a logical theorem of first-order logic that something exists — usually expressed as the assertion that something is identical to itself, ∃x(x=x). Consequently, it is a theorem of every first-order theory that something exists. However, as noted above, because in the intended semantics of ZFC there are only sets, the interpretation of this logical theorem in the context of ZFC is that some *set* exists. Hence, there is no need for a separate axiom asserting that a set exists. Second, however, even if ZFC is formulated in so-called free logic, in which it is not provable from logic alone that something exists, the axiom of infinity (below) asserts that an *infinite* set exists. This implies that *a* set exists and so, once again, it is superfluous to include an axiom asserting as much.

3.2.1 1. Axiom of extensionality

Main article: Axiom of extensionality

Two sets are equal (are the same set) if they have the same elements.

$$\forall x \forall y [\forall z (z \in x \Leftrightarrow z \in y) \Rightarrow x = y].$$

The converse of this axiom follows from the substitution property of equality. If the background logic does not include equality "=", *x*=*y* may be defined as an abbreviation for the following formula:[4]

$$\forall z [z \in x \Leftrightarrow z \in y] \wedge \forall w [x \in w \Leftrightarrow y \in w].$$

In this case, the axiom of extensionality can be reformulated as

$$\forall x \forall y [\forall z (z \in x \Leftrightarrow z \in y) \Rightarrow \forall w (x \in w \Leftrightarrow y \in w)],$$

which says that if *x* and *y* have the same elements, then they belong to the same sets.[5]

3.2.2 2. Axiom of regularity (also called the Axiom of foundation)

Main article: Axiom of regularity

Every non-empty set *x* contains a member *y* such that *x* and *y* are disjoint sets.

$$\forall x [\exists a (a \in x) \Rightarrow \exists y (y \in x \wedge \neg \exists z (z \in y \wedge z \in x))].$$ [6]

This implies, for example, that no set is an element of itself and that every set has an ordinal rank.

3.2.3 3. Axiom schema of specification (also called the axiom schema of separation or of restricted comprehension)

Main article: Axiom schema of specification

Subsets are commonly constructed using set builder notation. For example, the even integers can be constructed as the subset of the integers $\mathbb{Z}$ satisfying the congruence modulo predicate $x \equiv 0 \pmod 2$:

$$\{x \in \mathbb{Z} : x \equiv 0 \pmod 2\}.$$

In general, the subset of a set z obeying a formula ϕ (x) with one free variable x may be written as:

$$\{x \in z : \phi(x)\}.$$

The axiom schema of specification states that this subset always exists (it is an axiom *schema* because there is one axiom for each ϕ). Formally, let ϕbe any formula in the language of ZFC with all free variables among $x, z, w_1, \ldots, w_n$ (y is *not* free in ϕ). Then:

$$\forall z \forall w_1 \forall w_2 \ldots \forall w_n \exists y \forall x [x \in y \Leftrightarrow (x \in z \wedge \phi)].$$

Note that the axiom schema of specification can only construct subsets, and does not allow the construction of sets of the more general form:

$$\{x : \phi(x)\}.$$

This restriction is necessary to avoid Russell's paradox and its variants that accompany naive set theory with unrestricted comprehension.

In some other axiomatizations of ZF, this axiom is redundant in that it follows from the axiom schema of replacement and the axiom of the empty set.

On the other hand, the axiom of specification can be used to prove the existence of the empty set, denoted $\varnothing$, once at least one set is known to exist (see above). One way to do this is to use a property ϕwhich no set has. For example, if w is any existing set, the empty set can be constructed as

$$\varnothing = \{u \in w \mid (u \in u) \wedge \neg(u \in u)\}$$

Thus the axiom of the empty set is implied by the nine axioms presented here. The axiom of extensionality implies the empty set is unique (does not depend on w). It is common to make a definitional extension that adds the symbol $\varnothing$ to the language of ZFC.

3.2.4 4. Axiom of pairing

Main article: Axiom of pairing

If x and y are sets, then there exists a set which contains x and y as elements.

$$\forall x \forall y \exists z (x \in z \wedge y \in z).$$

The axiom schema of specification must be used to reduce this to a set with exactly these two elements. The axiom of pairing is part of Z, but is redundant in ZF because it follows from the axiom schema of replacement, if we are given a set with at least two elements. The existence of a set with at least two elements is assured by either the axiom of infinity, or by the axiom schema of specification and the axiom of the power set applied twice to any set.

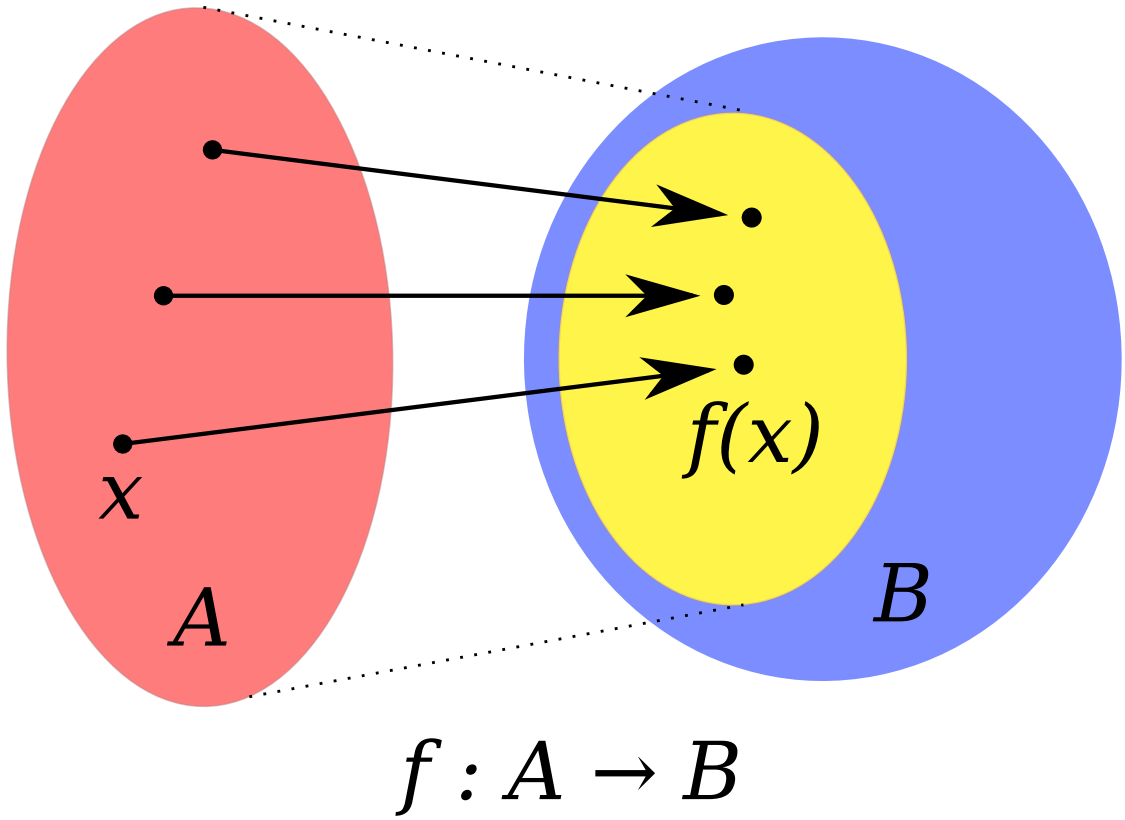
3.2.5 5. Axiom of union

Main article: Axiom of union

The union over the elements of a set exists. For example, the union over the elements of the set $\{\{1,2\},\{2,3\}\}$ is $\{1,2,3\}$.

Formally, for any set of sets $\mathcal{F}$ there is a set A containing every element that is a member of some member of $\mathcal{F}$:

$\forall \mathcal{F}\, \exists A\, \forall Y\, \forall x[(x \in Y \wedge Y \in \mathcal{F}) \Rightarrow x \in A].$



Axiom schema of replacement: the image of the domain set A *under the definable function* f *(i.e. the range of* f*) falls inside a set* B.

3.2.6 6. Axiom schema of replacement

Main article: Axiom schema of replacement

The axiom schema of replacement asserts that the image of a set under any definable function will also fall inside a set.

Formally, let ϕ be any formula in the language of ZFC whose free variables are among $x, y, A, w_1, \ldots, w_n$, so that in particular B is not free in ϕ. Then:

$\forall A \forall w_1 \forall w_2 \ldots \forall w_n \big[\forall x(x \in A \Rightarrow \exists! y\, \phi) \Rightarrow \exists B\, \forall x\big(x \in A \Rightarrow \exists y(y \in B \wedge \phi)\big)\big].$

In other words, if the relation ϕ represents a definable function f, A represents its domain, and $f(x)$ is a set for every x in that domain, then the range of f is a subset of some set B . The form stated here, in which B may be larger than strictly necessary, is sometimes called the axiom schema of collection.

3.2.7 7. Axiom of infinity

Main article: Axiom of infinity

Let $S(w)$ abbreviate $w \cup \{w\}$, where w is some set. (We can see that $\{w\}$ is a valid set by applying the Axiom of Pairing with $x = y = w$ so that the set z is $\{w\}$). Then there exists a set X such that the empty set $\varnothing$ is a member of X and, whenever a set y is a member of X, then $S(y)$ is also a member of X.

$$\exists X \left[\varnothing \in X \wedge \forall y(y \in X \Rightarrow S(y) \in X)\right].$$

More colloquially, there exists a set X having infinitely many members. (It must be established, however, that these members are all different, because if two elements are same, the sequence will loop around in a finite cycle of sets. The axiom of regularity prevents this from happening.) The minimal set X satisfying the axiom of infinity is the von Neumann ordinal ω, which can also be thought of as the set of natural numbers $\mathbb{N}$.

3.2.8 8. Axiom of power set

Main article: Axiom of power set

By definition a set z is a subset of a set x if and only if every element of z is also an element of x:

$$(z \subseteq x) \Leftrightarrow (\forall q(q \in z \Rightarrow q \in x)).$$

The Axiom of Power Set states that for any set x, there is a set y that contains every subset of x:

$$\forall x \exists y \forall z[z \subseteq x \Rightarrow z \in y].$$

The axiom schema of specification is then used to define the power set *P(x)* as the subset of such a y containing the subsets of x exactly:

$$P(x) = \{z \in y : z \subseteq x\}$$

Axioms **1–8** define ZF. Alternative forms of these axioms are often encountered, some of which are listed in Jech (2003). Some ZF axiomatizations include an axiom asserting that the empty set exists. The axioms of pairing, union, replacement, and power set are often stated so that the members of the set x whose existence is being asserted are just those sets which the axiom asserts x must contain.

The following axiom is added to turn ZF into ZFC:

3.2.9 9. Well-ordering theorem

Main article: Well-ordering theorem

For any set X, there is a binary relation R which well-orders X. This means R is a linear order on X such that every nonempty subset of X has a member which is minimal under R.

$$\forall X \exists R(R \text{ well-orders } X).$$

Given axioms **1–8**, there are many statements provably equivalent to axiom **9**, the best known of which is the axiom of choice (AC), which goes as follows. Let X be a set whose members are all non-empty. Then there exists a function f from X to the union of the members of X, called a "choice function", such that for all $Y \in X$ one has $f(Y) \in Y$. Since the existence of a choice function when X is a finite set is easily proved from axioms **1–8**, AC only matters for certain infinite sets. AC is characterized as nonconstructive because it asserts the existence of a choice set but says nothing about how the choice set is to be "constructed." Much research has sought to characterize the definability (or lack thereof) of certain sets whose existence AC asserts.

3.3 Motivation via the cumulative hierarchy

One motivation for the ZFC axioms is the cumulative hierarchy of sets introduced by John von Neumann.[7] In this viewpoint, the universe of set theory is built up in stages, with one stage for each ordinal number. At stage 0 there are no sets yet. At each following stage, a set is added to the universe if all of its elements have been added at previous stages. Thus the empty set is added at stage 1, and the set containing the empty set is added at stage 2.[8] The collection of all sets that are obtained in this way, over all the stages, is known as V. The sets in V can be arranged into a hierarchy by assigning to each set the first stage at which that set was added to V.

It is provable that a set is in V if and only if the set is pure and well-founded; and provable that V satisfies all the axioms of ZFC, if the class of ordinals has appropriate reflection properties. For example, suppose that a set x is added at stage α, which means that every element of x was added at a stage earlier than α. Then every subset of x is also added at stage α, because all elements of any subset of x were also added before stage α. This means that any subset of x which the axiom of separation can construct is added at stage α, and that the powerset of x will be added at the next stage after α. For a complete argument that V satisfies ZFC see Shoenfield (1977).

The picture of the universe of sets stratified into the cumulative hierarchy is characteristic of ZFC and related axiomatic set theories such as Von Neumann–Bernays–Gödel set theory (often called NBG) and Morse–Kelley set theory. The cumulative hierarchy is not compatible with other set theories such as New Foundations.

It is possible to change the definition of V so that at each stage, instead of adding all the subsets of the union of the previous stages, subsets are only added if they are definable in a certain sense. This results in a more "narrow" hierarchy which gives the constructible universe L, which also satisfies all the axioms of ZFC, including the axiom of choice. It is independent from the ZFC axioms whether $V = L$. Although the structure of L is more regular and well behaved than that of V, few mathematicians argue that $V = L$ should be added to ZFC as an additional axiom.

3.4 Metamathematics

The axiom schemata of replacement and separation each contain infinitely many instances. Montague (1961) included a result first proved in his 1957 Ph.D. thesis: if ZFC is consistent, it is impossible to axiomatize ZFC using only finitely many axioms. On the other hand, Von Neumann–Bernays–Gödel set theory (NBG) can be finitely axiomatized. The ontology of NBG includes proper classes as well as sets; a set is any class that can be a member of another class. NBG and ZFC are equivalent set theories in the sense that any theorem not mentioning classes and provable in one theory can be proved in the other.

Gödel's second incompleteness theorem says that a recursively axiomatizable system that can interpret Robinson arithmetic can prove its own consistency only if it is inconsistent. Moreover, Robinson arithmetic can be interpreted in general set theory, a small fragment of ZFC. Hence the consistency of ZFC cannot be proved within ZFC itself (unless it is actually inconsistent). Thus, to the extent that ZFC is identified with ordinary mathematics, the consistency of ZFC cannot be demonstrated in ordinary mathematics. The consistency of ZFC does follow from the existence of a weakly inaccessible cardinal, which is unprovable in ZFC if ZFC is consistent. Nevertheless, it is deemed unlikely that ZFC harbors an unsuspected contradiction; it is widely believed that if ZFC were inconsistent, that fact would have been uncovered by now. This much is certain — ZFC is immune to the classic paradoxes of naive set theory: Russell's paradox, the Burali-Forti paradox, and Cantor's paradox.

Abian & LaMacchia (1978) studied a subtheory of ZFC consisting of the axioms of extensionality, union, powerset, replacement, and choice. Using models, they proved this subtheory consistent, and proved that each of the axioms of extensionality, replacement, and power set is independent of the four remaining axioms of this subtheory. If this subtheory is augmented with the axiom of infinity, each of the axioms of union, choice, and infinity is independent of the five remaining axioms. Because there are non-well-founded models that satisfy each axiom of ZFC except the

axiom of regularity, that axiom is independent of the other ZFC axioms.

If consistent, ZFC cannot prove the existence of the inaccessible cardinals that category theory requires. Huge sets of this nature are possible if ZF is augmented with Tarski's axiom.[9] Assuming that axiom turns the axioms of infinity, power set, and choice (**7 – 9** above) into theorems.

3.4.1 Independence

Many important statements are independent of ZFC (see list of statements undecidable in ZFC). The independence is usually proved by forcing, whereby it is shown that every countable transitive model of ZFC (sometimes augmented with large cardinal axioms) can be expanded to satisfy the statement in question. A different expansion is then shown to satisfy the negation of the statement. An independence proof by forcing automatically proves independence from arithmetical statements, other concrete statements, and large cardinal axioms. Some statements independent of ZFC can be proven to hold in particular inner models, such as in the constructible universe. However, some statements that are true about constructible sets are not consistent with hypothesized large cardinal axioms.

Forcing proves that the following statements are independent of ZFC:

- Continuum hypothesis
- Diamond principle
- Suslin hypothesis
- Martin's axiom (which is not a ZFC axiom)
- Axiom of Constructibility (V=L) (which is also not a ZFC axiom).

Remarks:

- The consistency of V=L is provable by inner models but not forcing: every model of ZF can be trimmed to become a model of ZFC + V=L.
- The Diamond Principle implies the Continuum Hypothesis and the negation of the Suslin Hypothesis.
- Martin's axiom plus the negation of the Continuum Hypothesis implies the Suslin Hypothesis.
- The constructible universe satisfies the Generalized Continuum Hypothesis, the Diamond Principle, Martin's Axiom and the Kurepa Hypothesis.
- The failure of the Kurepa hypothesis is equiconsistent with the existence of a strongly inaccessible cardinal.

A variation on the method of forcing can also be used to demonstrate the consistency and unprovability of the axiom of choice, i.e., that the axiom of choice is independent of ZF. The consistency of choice can be (relatively) easily verified by proving that the inner model L satisfies choice. (Thus every model of ZF contains a submodel of ZFC, so that Con(ZF) implies Con(ZFC).) Since forcing preserves choice, we cannot directly produce a model contradicting choice from a model satisfying choice. However, we can use forcing to create a model which contains a suitable submodel, namely one satisfying ZF but not C.

Another method of proving independence results, one owing nothing to forcing, is based on Gödel's second incompleteness theorem. This approach employs the statement whose independence is being examined, to prove the existence of a set model of ZFC, in which case Con(ZFC) is true. Since ZFC satisfies the conditions of Gödel's second theorem, the consistency of ZFC is unprovable in ZFC (provided that ZFC is, in fact, consistent). Hence no statement allowing such a proof can be proved in ZFC. This method can prove that the existence of large cardinals is not provable in ZFC, but cannot prove that assuming such cardinals, given ZFC, is free of contradiction.

3.5 Criticisms

For criticism of set theory in general, see Objections to set theory

ZFC has been criticized both for being excessively strong and for being excessively weak, as well as for its failure to capture objects such as proper classes and the universal set.

Many mathematical theorems can be proven in much weaker systems than ZFC, such as Peano arithmetic and second order arithmetic (as explored by the program of reverse mathematics). Saunders Mac Lane and Solomon Feferman have both made this point. Some of "mainstream mathematics" (mathematics not directly connected with axiomatic set theory) is beyond Peano arithmetic and second order arithmetic, but still, all such mathematics can be carried out in ZC (Zermelo set theory with choice), another theory weaker than ZFC. Much of the power of ZFC, including the axiom of regularity and the axiom schema of replacement, is included primarily to facilitate the study of the set theory itself.

On the other hand, among axiomatic set theories, ZFC is comparatively weak. Unlike New Foundations, ZFC does not admit the existence of a universal set. Hence the universe of sets under ZFC is not closed under the elementary operations of the algebra of sets. Unlike von Neumann–Bernays–Gödel set theory and Morse–Kelley set theory (MK), ZFC does not admit the existence of proper classes. A further comparative weakness of ZFC is that the axiom of choice included in ZFC is weaker than the axiom of global choice included in MK.

There are numerous mathematical statements undecidable in ZFC. These include the continuum hypothesis, the Whitehead problem, and the Normal Moore space conjecture. Some of these conjectures are provable with the addition of axioms such as Martin's axiom, large cardinal axioms to ZFC. Some others are decided in ZF+AD where AD is the axiom of determinacy, a strong supposition incompatible with choice. One attraction of large cardinal axioms is that they enable many results from ZF+AD to be established in ZFC adjoined by some large cardinal axiom (see projective determinacy). The Mizar system and Metamath have adopted Tarski–Grothendieck set theory, an extension of ZFC, so that proofs involving Grothendieck universes (encountered in category theory and algebraic geometry) can be formalized.

3.6 See also

- Foundation of mathematics
- Inner model
- Large cardinal axiom

Related axiomatic set theories:

- Morse–Kelley set theory
- Von Neumann–Bernays–Gödel set theory
- Tarski–Grothendieck set theory
- Constructive set theory
- Internal set theory

3.7 Notes

[1] Ciesielski 1997.

[2] Ebbinghaus 2007, p. 136.

[3] Kunen (1980), p. 10.

[4] Hatcher 1982, p. 138, def. 1.

[5] Fraenkel, Bar-Hillel & Lévy 1973.

[6] Shoenfield 2001, p. 239.

[7] Shoenfield 1977, section 2.

[8] Hinman 2005, p. 467.

[9] Tarski 1939.

3.8 References

- Abian, Alexander (1965). *The Theory of Sets and Transfinite Arithmetic*. W B Saunders.
- ———; LaMacchia, Samuel (1978). "On the Consistency and Independence of Some Set-Theoretical Axioms". *Notre Dame Journal of Formal Logic* **19**: 155–58. doi:10.1305/ndjfl/1093888220.
- Ciesielski, Krzysztof (1997). *Set Theory for the Working Mathematician*. Cambridge University Press. p. 4. ISBN 0-521-59441-3. Retrieved 12 August 2015.
- Devlin, Keith (1996) [1984]. *The Joy of Sets*. Springer.
- Ebbinghaus, Heinz-Dieter (2007). *Ernst Zermelo: An Approach to His Life and Work*. Springer. ISBN 978-3-540-49551-2.
- Fraenkel, Abraham; Bar-Hillel, Yehoshua; Lévy, Azriel (1973) [1958]. *Foundations of Set Theory*. North-Holland. Fraenkel's final word on ZF and ZFC.
- Hatcher, William (1982) [1968]. *The Logical Foundations of Mathematics*. Pergamon Press.
- Hinman, Peter (2005). *Fundamentals of Mathematical Logic*. A K Peters. ISBN 978-1-56881-262-5.
- Jech, Thomas (2003). *Set Theory: The Third Millennium Edition, Revised and Expanded*. Springer. ISBN 3-540-44085-2.
- Kunen, Kenneth (1980). *Set Theory: An Introduction to Independence Proofs*. Elsevier. ISBN 0-444-86839-9.
- Montague, Richard (1961). "Semantic closure and non-finite axiomatizability". *Infinistic Methods*. London: Pergamon Press. pp. 45–69.
- Shoenfield, Joseph R. (2001) [1967]. *Mathematical Logic* (2nd ed.). A K Peters. ISBN 978-1-56881-135-2.
- Shoenfield, Joseph R. (1977). "Axioms of set theory". In Barwise, K. J. *Handbook of Mathematical Logic*. ISBN 0-7204-2285-X.
- Patrick Suppes, 1972 (1960). *Axiomatic Set Theory*. Dover reprint. Perhaps the best exposition of ZFC before the independence of AC and the Continuum hypothesis, and the emergence of large cardinals. Includes many theorems.
- Gaisi Takeuti and Zaring, W M, 1971. *Introduction to Axiomatic Set Theory*. Springer-Verlag.
- Tarski, Alfred (1939). "On well-ordered subsets of any set". *Fundamenta Mathematicae* **32**: 176–83.
- Tiles, Mary, 2004 (1989). *The Philosophy of Set Theory*. Dover reprint. Weak on metatheory; the author is not a mathematician.
- Tourlakis, George, 2003. *Lectures in Logic and Set Theory, Vol. 2*. Cambridge University Press.
- Jean van Heijenoort, 1967. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Harvard University Press. Includes annotated English translations of the classic articles by Zermelo, Fraenkel, and Skolem bearing on **ZFC**.
- Zermelo, Ernst (1908). "Untersuchungen über die Grundlagen der Mengenlehre I". *Mathematische Annalen* **65**: 261–281. doi:10.1007/BF01449999. English translation in Heijenoort, Jean van (1967). "Investigations in the foundations of set theory". *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Source Books in the History of the Sciences. Harvard University Press. pp. 199–215. ISBN 978-0-674-32449-7.
- Zermelo, Ernst (1930). "Über Grenzzahlen und Mengenbereiche". *Fundamenta Mathematicae* **16**: 29–47. ISSN 0016-2736.

3.9 External links

- Hazewinkel, Michiel, ed. (2001), "ZFC", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Stanford Encyclopedia of Philosophy articles by Thomas Jech:
  - Set Theory;
  - Axioms of Zermelo–Fraenkel Set Theory.
- Metamath version of the ZFC axioms — A concise and nonredundant axiomatization. The background first order logic is defined especially to facilitate machine verification of proofs.
  - A derivation in Metamath of a version of the separation schema from a version of the replacement schema.
- Zermelo-Fraenkel Axioms at PlanetMath.org.
- Weisstein, Eric W., "Zermelo-Fraenkel Set Theory", *MathWorld*.

Chapter 4

Presburger arithmetic

Presburger arithmetic is the first-order theory of the natural numbers with addition, named in honor of Mojżesz Presburger, who introduced it in 1929. The signature of Presburger arithmetic contains only the addition operation and equality, omitting the multiplication operation entirely. The axioms include a schema of induction.

Presburger arithmetic is much weaker than Peano arithmetic, which includes both addition and multiplication operations. Unlike Peano arithmetic, Presburger arithmetic is a decidable theory. This means it is possible to algorithmically determine, for any sentence in the language of Presburger arithmetic, whether that sentence is provable from the axioms of Presburger arithmetic. The asymptotic running-time computational complexity of this decision problem is doubly exponential, however, as shown by Fischer & Rabin (1974).

4.1 Overview

The language of Presburger arithmetic contains constants 0 and 1 and a binary function +, interpreted as addition. In this language, the axioms of Presburger arithmetic are the universal closures of the following:

1. $\neg(0 = x + 1)$
2. $x + 1 = y + 1 \to x = y$
3. $x + 0 = x$
4. $x + (y + 1) = (x + y) + 1$
5. Let $P(x)$ be a first-order formula in the language of Presburger arithmetic with a free variable x (and possibly other free variables). Then the following formula is an axiom:

$$(P(0) \wedge \forall x(P(x) \to P(x + 1))) \to \forall y\, P(y).$$

(5) is an axiom schema of induction, representing infinitely many axioms. Since the axioms in the schema in (5) cannot be replaced by any finite number of axioms, Presburger arithmetic is not finitely axiomatizable in first-order logic.

Presburger arithmetic cannot formalize concepts such as divisibility or prime number. Generally, any number concept leading to multiplication cannot be defined in Presburger arithmetic, since that leads to incompleteness and undecidability. However, it can formulate individual instances of divisibility; for example, it proves "for all x, there exists $y : (y + y = x) \vee (y + y + 1 = x)$". This states that every number is either even or odd.

4.2 Properties

Mojżesz Presburger proved Presburger arithmetic to be:

- consistent: There is no statement in Presburger arithmetic which can be deduced from the axioms such that its negation can also be deduced.
- complete: For each statement in the language of Presburger arithmetic, either it is possible to deduce it from the axioms or it is possible to deduce its negation.
- decidable: There exists an algorithm which decides whether any given statement in Presburger arithmetic is a theorem or a nontheorem.

The decidability of Presburger arithmetic can be shown using quantifier elimination, supplemented by reasoning about arithmetical congruence (Enderton 2001, p. 188).

Peano arithmetic, which is Presburger arithmetic augmented with multiplication, is not decidable, as a consequence of the negative answer to the Entscheidungsproblem. By Gödel's incompleteness theorem, Peano arithmetic is incomplete and its consistency is not internally provable (but see Gentzen's consistency proof).

The decision problem for Presburger arithmetic is an interesting example in computational complexity theory and computation. Let n be the length of a statement in Presburger arithmetic. Then Fischer and Rabin (1974) proved that any decision algorithm for Presburger arithmetic has a worst-case runtime of at least $2^{2^{cn}}$, for some constant c>0. Hence, the decision problem for Presburger arithmetic is an example of a decision problem that has been proved to require more than exponential run time. Fischer and Rabin also proved that for any reasonable axiomatization (defined precisely in their paper), there exist theorems of length n which have doubly exponential length proofs. Intuitively, this means there are computational limits on what can be proven by computer programs. Fischer and Rabin's work also implies that Presburger arithmetic can be used to define formulas which correctly calculate any algorithm as long as the inputs are less than relatively large bounds. The bounds can be increased, but only by using new formulas. On the other hand, a triply exponential upper bound on a decision procedure for Presburger Arithmetic was proved by Oppen (1978). A more tight complexity bound was shown using alternating complexity classes by Berman (1980).

4.3 Applications

Because Presburger arithmetic is decidable, automatic theorem provers for Presburger arithmetic exist. For example, the Coq proof assistant system features the tactic omega for Presburger arithmetic and the Isabelle proof assistant contains a verified quantifier elimination procedure by Nipkow (2010). The double exponential complexity of the theory makes it infeasible to use the theorem provers on complicated formulas, but this behavior occurs only in the presence of nested quantifiers: Oppen and Nelson (1980) describe an automatic theorem prover which uses the simplex algorithm on an extended Presburger arithmetic without nested quantifiers to prove some of the instances of quantifier-free Presburger arithmetic formulas. More recent Satisfiability Modulo Theories solvers use complete integer programming techniques to handle quantifier-free fragment of Presburger arithmetic theory (King, Barrett, Tinelli 2014).

Presburger arithmetic can be extended to include multiplication by constants, since multiplication is repeated addition. Most array subscript calculations then fall within the region of decidable problems. This approach is the basis of at least five proof-of-correctness systems for computer programs, beginning with the Stanford Pascal Verifier in the late 1970s and continuing through to Microsoft's Spec# system of 2005.

4.4 Presburger-definable integer relation

Some properties are now given about integer relations definable in Presburger Arithmetic. For the sake of simplicity, all relations considered in this sections are over natural integers.

A relation is Presburger-definable if and only if it is a semilinear set.[1]

A unary integer relation R , that is, a set of natural integers, is Presburger-definable if and only if it is ultimately periodic. That is, if there exists a threshold $t \in \mathbb{N}$ and a positive period $p \in \mathbb{N}^{>0}$ such that, for all integer n such that $|n| \geq t$, $n \in R$ if and only if $n + p \in R$.

By the Cobham–Semenov theorem, a relation is Presburger-definable if and only if it is definable in Büchi arithmetic of base k for all $k \geq 2$.[2][3] A relation definable in Büchi aritmetic of base k and k' for k and k' being multiplicatively independent integers is Presburger definable.

An integer relation R is Presburger-definable if and only if all sets of integers which are definable in first order logic with addition and R (that is, Presburger Arithmetic plus a predicate for R) are Presburger-definable.[4] Equivalently, for each relation R which is not Presburger-definable, there exists a first-order formula with addition and R which defines a set of integer which is not definable using only addition.

4.4.1 Muchnik's characterization

Presburger-definable relations admit another characterization: by Muchnik's theorem.[5] It is more complicated to state, but led to the proof of the two former characterizations. Before Muchnik's theorem can be stated, some additional definitions must be introduced.

For a set $R \subseteq \mathbb{N}^d$, the section $x_i = j$ of R , for $i < d$ and $j \in \mathbb{N}$ is defined as $\{(x_0, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{d-1}) \in \mathbb{N}^{d-1} \mid (x_0, \ldots, x_{i-1}, j, x_{i+1}, \ldots, x_{d-1}) \in R\}$.

Given two sets $R, S \subseteq \mathbb{N}^d$ and a d -tuple of integers $(p_0, \ldots, p_{d-1}) \in \mathbb{N}^d$, the set R is called $(p_0, \ldots, p_{d-1})$ -periodic in S if, for all $(x_0, \ldots, x_{d-1}) \in S$ such that $(x_0 + p_0, \ldots, x_{d-1} + p_{d-1}) \in S$, then $(x_0, \ldots, x_{d-1}) \in R$ if and only if $(x_0 + p_0, \ldots, x_{d-1} + p_{d-1}) \in R$. For $s \in \mathbb{N}$, the set R is said to be s -periodic in S if it is $(p_0, \ldots, p_{d-1})$ -periodic for some $(p_0, \ldots, p_{d-1}) \in \mathbb{Z}^d$ such that $\sum_{i=0}^{d-1} |p_i| < s$.

Finally, for $k, x_0, \ldots, x_{d-1} \in \mathbb{N}$, let $C(k, (x_0, \ldots, x_{d-1})) = \{(x_0 + c_0, \ldots, x_{d-1} + c_{d-1}) \mid 0 \leq c_i < k\}$ denote the cube of size k whose lesser corner is $(x_0, \ldots, x_{d-1})$.

Muchnik's theorem then says that a set $R \subseteq \mathbb{N}^d$ is Presburger-definable if and only if:

- if the dimension d is greater than 1 then all sections of R are Presburger-definable and
- there exists $s \in \mathbb{N}$ such that, for every $k \in \mathbb{N}$, there exists $t \in \mathbb{N}$ such that for all $(x_0, \ldots, x_{d-1}) \in \mathbb{N}^d$ with $\sum_{i=0}^{d-1} x_i > t$, R is s -periodic in $C(k, (x_0, \ldots, x_{d-1}))$.

Intuitively, the integer s represents the length of a shift, the integer k is the size of the cubes and t is the threshold before the periodicity. This result remains true when the condition $\sum_{i=0}^{d-1} x_i > t$ is replaced either by $\min(x_0, \ldots, x_{d-1}) > t$ or by $\max(x_0, \ldots, x_{d-1}) > t$.

This characterization led to the so-called "definable criterion for definability in Presburger arithmetic", that is: there exists a first-order formula with addition and a d -ary predicate R which holds if and only if R is interpreted by a Presburger-definable relation. Muchnik's theorem also allows one to prove that it is decidable whether an automatic sequence accepts a Presburger-definable set.

4.5 See also

- Robinson arithmetic

4.6 References

[1] Ginsburg, Seymour; Spanier, Edwin Henry (1966). "Semigroups, Presburger Formulas, and Languages". *Pacific Journal of Mathematics* **16**: 285–296.

[2] Cobham, Alan (1969). "On the base-dependence of sets of numbers recognizable by finite automata". *Math. Systems Theory* **3**: 186–192. doi:10.1007/BF01746527.

[3] Semenov, A. L. (1977). "Presburgerness of predicates regular in two number systems". *Sibirsk. Mat. Zh.* (in Russian) **18**: 403–418.

[4] Michaux, Christian; Villemaire, Roger (1996). "Presburger Arithmetic and Recognizability of Sets of Natural Numbers by Automata: New Proofs of Cobham's and Semenov's Theorems". *Ann. Pure Appl. Logic* **77**: 251–277. doi:10.1016/0168-0072(95)00022-4.

[5] Muchnik, Andrei A. (2003). "The definable criterion for definability in Presburger arithmetic and its applications". *Theor. Comput. Sci.* **290**: 1433–1444. doi:10.1016/S0304-3975(02)00047-6.

- Cooper, D. C., 1972, "Theorem Proving in Arithmetic without Multiplication" in B. Meltzer and D. Michie, eds., *Machine Intelligence Vol. 7*. Edinburgh University Press: 91–99.
- Enderton, Herbert (2001). *A mathematical introduction to logic* (2nd ed.). Boston, MA: Academic Press. ISBN 978-0-12-238452-3.
- Ferrante, Jeanne, and Charles W. Rackoff, 1979. *The Computational Complexity of Logical Theories*. Lecture Notes in Mathematics 718. Springer-Verlag.
- Fischer, Michael J.; Rabin, Michael O. (1974). "Super-Exponential Complexity of Presburger Arithmetic". *Proceedings of the SIAM-AMS Symposium in Applied Mathematics* **7**: 27–41.
- G. Nelson and D. C. Oppen (Apr 1978). "A simplifier based on efficient decision algorithms". *Proc. 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*: 141–150. doi:10.1145/512760.512775.
- Mojżesz Presburger, 1929, "Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt" in *Comptes Rendus du I congrès de Mathématiciens des Pays Slaves*. Warszawa: 92–101. — see Stansifer (1984)for an English translation
- Ryan Stansifer (Sep 1984). Presburger's Article on Integer Arithmetic: Remarks and Translation (PDF) (Technical Report). TR84-639. Ithaca/NY: Dept. of Computer Science, Cornell University.
- William Pugh, 1991, "The Omega test: a fast and practical integer programming algorithm for dependence analysis,".
- Reddy, C. R., and D. W. Loveland, 1978, "Presburger Arithmetic with Bounded Quantifier Alternation." *ACM Symposium on Theory of Computing*: 320–325.
- Young, P., 1985, "Gödel theorems, exponential difficulty and undecidability of arithmetic theories: an exposition" in A. Nerode and R. Shore, Recursion Theory, American Mathematical Society: 503-522.
- Oppen, Derek C. (1978). "A $2^{2^{2^{pn}}}$ Upper Bound on the Complexity of Presburger Arithmetic" (PDF). *J. Comput. Syst. Sci.* **16** (3): 323–332. doi:10.1016/0022-0000(78)90021-1.
- Berman, L. (1980). "The Complexity of Logical Theories". *Theoretical Computer Science* **11** (1): 71–77. doi:10.1016/0304-3975(80)90037-7.
- Nipkow, T (2010). "Linear Quantifier Elimination". *Journal of Automated Reasoning* **45** (2): 189–212. doi:10.1007/s10817-010-9183-0.
- King, Tim; Barrett, Clark W.; Tinelli, Cesare (2014). "Leveraging linear and mixed integer programming for SMT". *FMCAD* **2014**: 139–146. doi:10.1109/FMCAD.2014.6987606.

4.7 External links

- A complete Theorem Prover for Presburger Arithmetic by Philipp Rümmer

Chapter 5

Lambda calculus

Lambda calculus (also written as **λ-calculus**) is a formal system in mathematical logic for expressing computation based on function abstraction and application using variable binding and substitution. It is a universal model of computation that can be used to simulate any single-taped Turing machine and was first introduced by mathematician Alonzo Church in the 1930s as part of an investigation into the foundations of mathematics.

5.1 Explanation and applications

Lambda calculus is Turing complete, that is, it is a universal model of computation that can be used to simulate any single-taped Turing machine.[1] Its namesake, the Greek letter lambda (λ), is used in **lambda expressions** and **lambda terms** to denote binding a variable in a function.

Lambda calculus may be *typed* and *untyped*. In typed lambda calculus functions can be applied only if they are capable of accepting the given input's "type" of data.

Lambda calculus has applications in many different areas in mathematics, philosophy,[2] linguistics,[3][4] and computer science.[5] Lambda calculus has played an important role in the development of the theory of programming languages. Functional programming languages implement the lambda calculus. Lambda calculus also is a current research topic in Category theory.[6]

5.2 Lambda calculus in history of mathematics

The lambda calculus was introduced by mathematician Alonzo Church in the 1930s as part of an investigation into the foundations of mathematics.[7][8] The original system was shown to be logically inconsistent in 1935 when Stephen Kleene and J. B. Rosser developed the Kleene–Rosser paradox.

Subsequently, in 1936 Church isolated and published just the portion relevant to computation, what is now called the untyped lambda calculus.[9] In 1940, he also introduced a computationally weaker, but logically consistent system, known as the simply typed lambda calculus.[10]

Until the 1960s when its relation to programming languages was clarified, the λ-calculus was only a formalism. Thanks to Montague and other linguists' applications in the semantics of natural language, the λ-calculus has begun to enjoy a respectable place in linguistics (see Heim and Kratzer 1998) and computer science, too.[11]

5.3 Informal description

5.3.1 Motivation

Computable functions are a fundamental concept within computer science and mathematics. The λ-calculus provides a simple semantics for computation, enabling properties of computation to be studied formally. The λ-calculus

incorporates two simplifications that make this semantics simple. The first simplification is that the λ-calculus treats functions "anonymously", without giving them explicit names. For example, the function

$\text{square\_sum}(x, y) = x^2 + y^2$

can be rewritten in *anonymous form* as

$(x, y) \mapsto x^2 + y^2$

(read as "the pair of x and y is mapped to $x^2 + y^2$ "). Similarly,

$\text{id}(x) = x$

can be rewritten in anonymous form as $x \mapsto x$, where the input is simply mapped to itself.

The second simplification is that the λ-calculus only uses functions of a single input. An ordinary function that requires two inputs, for instance the square_sum function, can be reworked into an equivalent function that accepts a single input, and as output returns *another* function, that in turn accepts a single input. For example,

$(x, y) \mapsto x^2 + y^2$

can be reworked into

$x \mapsto (y \mapsto x^2 + y^2)$

This method, known as currying, transforms a function that takes multiple arguments into a chain of functions each with a single argument.

Function application of the square_sum function to the arguments (5, 2), yields at once

$((x, y) \mapsto x^2 + y^2)(5, 2)$

$= 5^2 + 2^2$

$= 29$

whereas evaluation of the curried version requires one more step

$((x \mapsto (y \mapsto x^2 + y^2))(5))(2)$

$= (y \mapsto 5^2 + y^2)(2)$

$= 5^2 + 2^2$

$= 29$

to arrive at the same result.

5.3.2 The lambda calculus

The lambda calculus consists of a language of **lambda terms**, which is defined by a certain formal syntax, and a set of transformation rules, which allow manipulation of the lambda terms. These transformation rules can be viewed as an equational theory or as an operational definition.

As described above, all functions in the lambda calculus are anonymous functions, having no names. They only accept one input variable, with currying used to implement functions with several variables.

Lambda terms

The syntax of the lambda calculus defines some expressions as valid lambda calculus expressions and some as invalid, just as some strings of characters are valid C programs and some are not. A valid lambda calculus expression is called a "lambda term".

The following three rules give an inductive definition that can be applied to build all syntactically valid lambda terms:

- a variable, x , is itself a valid lambda term
- if t is a lambda term, and x is a variable, then $(\lambda x.t)$ is a lambda term (called a **lambda abstraction**);
- if t and s are lambda terms, then (ts) is a lambda term (called an **application**).

Nothing else is a lambda term. Thus a lambda term is valid if and only if it can be obtained by repeated application of these three rules. However, some parentheses can be omitted according to certain rules. For example, the outermost parentheses are usually not written. See *Notation*, below.

A **lambda abstraction** $\lambda x.t$ is a definition of an anonymous function that is capable of taking a single input x and substituting it into the expression t . It thus defines an anonymous function that takes x and returns t . For example, $\lambda x.x^2 + 2$ is a lambda abstraction for the function $f(x) = x^2 + 2$ using the term $x^2 + 2$ for t . The definition of a function with a lambda abstraction merely "sets up" the function but does not invoke it. The abstraction binds the variable x in the term t .

An **application** ts represents the application of a function t to an input s , that is, it represents the act of calling function t on input s to produce $t(s)$.

There is no concept in lambda calculus of variable declaration. In a definition such as $\lambda x.x + y$ (i.e. $f(x) = x + y$), the lambda calculus treats y as a variable that is not yet defined. The lambda abstraction $\lambda x.x + y$ is syntactically valid, and represents a function that adds its input to the yet-unknown y .

Bracketing may be used and may be needed to disambiguate terms. For example, $\lambda x.((\lambda x.x)x)$ and $(\lambda x.(\lambda x.x))x$ denote different terms (although they coincidentally reduce to the same value). Here the first example defines a function that defines a function and returns the result of applying x to the child-function (apply function then return), while the second example defines a function that returns a function for any input and then returns it on application of x (return function then apply).

Functions that operate on functions

In lambda calculus, functions are taken to be 'first class values', so functions may be used as the inputs, or be returned as outputs from other functions.

For example, $\lambda x.x$ represents the identity function, $x \mapsto x$, and $(\lambda x.x)y$ represents the identity function applied to y . Further, $(\lambda x.y)$ represents the **constant function** $x \mapsto y$, the function that always returns y , no matter the input. In lambda calculus, function application is regarded as left-associative, so that stx means $(st)x$.

There are several notions of "equivalence" and "reduction" that allow lambda terms to be "reduced" to "equivalent" lambda terms.

Alpha equivalence

A basic form of equivalence, definable on lambda terms, is alpha equivalence. It captures the intuition that the particular choice of a bound variable, in a lambda abstraction, does not (usually) matter. For instance, $\lambda x.x$ and $\lambda y.y$ are alpha-equivalent lambda terms, and they both represent the same function (the identity function). The terms x and y are not alpha-equivalent, because they are not bound in a lambda abstraction. In many presentations, it is usual to identify alpha-equivalent lambda terms.

The following definitions are necessary in order to be able to define beta reduction:

Free variables

The **free variables** of a term are those variables not bound by a lambda abstraction. The set of free variables of an expression is defined inductively:

- The free variables of x are just x
- The set of free variables of $\lambda x.t$ is the set of free variables of t , but with x removed
- The set of free variables of ts is the union of the set of free variables of t and the set of free variables of s .

For example, the lambda term representing the identity $\lambda x.x$ has no free variables, but the function $\lambda x.yx$ has a single free variable, y .

Capture-avoiding substitutions

Suppose t , s and r are lambda terms and x and y are variables. The notation $t[x := r]$ indicates substitution of r for x in t in a *capture-avoiding* manner. This is defined so that:

- $x[x := r] = r$;
- $y[x := r] = y$ if $x \neq y$;
- $(ts)[x := r] = (t[x := r])(s[x := r])$;
- $(\lambda x.t)[x := r] = \lambda x.t$;
- $(\lambda y.t)[x := r] = \lambda y.(t[x := r])$ if $x \neq y$ and y is not in the free variables of r . The variable y is said to be "fresh" for r .

For example, $(\lambda x.x)[y := y] = \lambda x.(x[y := y]) = \lambda x.x$, and $((\lambda x.y)x)[x := y] = ((\lambda x.y)[x := y])(x[x := y]) = (\lambda x.y)y$.

The freshness condition (requiring that y is not in the free variables of r) is crucial in order to ensure that substitution does not change the meaning of functions. For example, a substitution is made that ignores the freshness condition: $(\lambda x.y)[y := x] = \lambda x.(y[y := x]) = \lambda x.x$. This substitution turns the constant function $\lambda x.y$ into the identity $\lambda x.x$ by substitution.

In general, failure to meet the freshness condition can be remedied by alpha-renaming with a suitable fresh variable. For example, switching back to our correct notion of substitution, in $(\lambda x.y)[y := x]$ the lambda abstraction can be renamed with a fresh variable z , to obtain $(\lambda z.y)[y := x] = \lambda z.(y[y := x]) = \lambda z.x$, and the meaning of the function is preserved by substitution.

Beta reduction

The beta reduction rule states that an application of the form $(\lambda x.t)s$ reduces to the term $t[x := s]$. The notation $(\lambda x.t)s \to t[x := s]$ is used to indicate that $(\lambda x.t)s$ beta reduces to $t[x := s]$. For example, for every s , $(\lambda x.x)s \to x[x := s] = s$. This demonstrates that $\lambda x.x$ really is the identity. Similarly, $(\lambda x.y)s \to y[x := s] = y$, which demonstrates that $\lambda x.y$ is a constant function.

The lambda calculus may be seen as an idealised functional programming language, like Haskell or Standard ML. Under this view, beta reduction corresponds to a computational step. This step can be repeated by additional beta conversions until there are no more applications left to reduce. In the untyped lambda calculus, as presented here, this reduction process may not terminate. For instance, consider the term $\Omega = (\lambda x.xx)(\lambda x.xx)$. Here $(\lambda x.xx)(\lambda x.xx) \to (xx)[x := \lambda x.xx] = (x[x := \lambda x.xx])(x[x := \lambda x.xx]) = (\lambda x.xx)(\lambda x.xx)$. That is, the term reduces to itself in a single beta reduction, and therefore the reduction process will never terminate.

Another aspect of the untyped lambda calculus is that it does not distinguish between different kinds of data. For instance, it may be desirable to write a function that only operates on numbers. However, in the untyped lambda calculus, there is no way to prevent a function from being applied to truth values, strings, or other non-number objects.

5.4 Formal definition

Main article: Lambda calculus definition

5.4.1 Definition

Lambda expressions are composed of:

- variables v_1, v_2, ..., v_n, ...
- the abstraction symbols lambda 'λ' and dot '.'
- parentheses ( )

The set of lambda expressions, Λ, can be defined inductively:

1. If x is a variable, then x ∈ Λ
2. If x is a variable and M ∈ Λ, then (λx.M) ∈ Λ
3. If M, N ∈ Λ, then (M N) ∈ Λ

Instances of rule 2 are known as abstractions and instances of rule 3 are known as applications.[12]

5.4.2 Notation

To keep the notation of lambda expressions uncluttered, the following conventions are usually applied:

- Outermost parentheses are dropped: M N instead of (M N)
- Applications are assumed to be left associative: M N P may be written instead of ((M N) P)[13]
- The body of an abstraction extends as far right as possible: λx.M N means λx.(M N) and not (λx.M) N
- A sequence of abstractions is contracted: λx.λy.λz.N is abbreviated as λxyz.N[14][15]

5.4.3 Free and bound variables

The abstraction operator, λ, is said to bind its variable wherever it occurs in the body of the abstraction. Variables that fall within the scope of an abstraction are said to be *bound*. All other variables are called *free*. For example, in the following expression y is a bound variable and x is free: $\lambda y.x\ x\ y$. Also note that a variable is bound by its "nearest" abstraction. In the following example the single occurrence of x in the expression is bound by the second lambda: $\lambda x.y\ (\lambda x.z\ x)$

The set of *free variables* of a lambda expression, M, is denoted as FV(M) and is defined by recursion on the structure of the terms, as follows:

1. FV(x) = {x}, where x is a variable
2. FV(λx.M) = FV(M) \ {x}
3. FV(M N) = FV(M) ∪ FV(N)[16]

An expression that contains no free variables is said to be *closed*. Closed lambda expressions are also known as combinators and are equivalent to terms in combinatory logic.

5.5 Reduction

The meaning of lambda expressions is defined by how expressions can be reduced.[17]

There are three kinds of reduction:

- **α-conversion**: changing bound variables (**alpha**);
- **β-reduction**: applying functions to their arguments (**beta**);
- **η-conversion**: which captures a notion of extensionality (**eta**).

We also speak of the resulting equivalences: two expressions are *β-equivalent*, if they can be β-converted into the same expression, and α/η-equivalence are defined similarly.

The term *redex*, short for *reducible expression*, refers to subterms that can be reduced by one of the reduction rules. For example, (λ*x*.M) N is a beta-redex in expressing the substitution of N for x in M; if *x* is not free in M, λ*x*.M *x* is also an eta-redex. The expression to which a redex reduces is called its reduct; using the previous example, the reducts of these expressions are respectively M[*x*:=N] and M.

5.5.1 α-conversion

Alpha-conversion, sometimes known as alpha-renaming,[18] allows bound variable names to be changed. For example, alpha-conversion of λ*x*.*x* might yield λ*y*.*y*. Terms that differ only by alpha-conversion are called *α-equivalent*. Frequently, in uses of lambda calculus, α-equivalent terms are considered to be equivalent.

The precise rules for alpha-conversion are not completely trivial. First, when alpha-converting an abstraction, the only variable occurrences that are renamed are those that are bound to the same abstraction. For example, an alpha-conversion of λ*x*.λ*x*.*x* could result in λ*y*.λ*x*.*x*, but it could *not* result in λ*y*.λ*x*.*y*. The latter has a different meaning from the original.

Second, alpha-conversion is not possible if it would result in a variable getting captured by a different abstraction. For example, if we replace *x* with *y* in λ*x*.λ*y*.*x*, we get λ*y*.λ*y*.*y*, which is not at all the same.

In programming languages with static scope, alpha-conversion can be used to make name resolution simpler by ensuring that no variable name masks a name in a containing scope (see alpha renaming to make name resolution trivial).

In the De Bruijn index notation, any two alpha-equivalent terms are literally identical.

Substitution

Substitution, written $E[V := R]$, is the process of replacing all free occurrences of the variable V in the expression E with expression R. Substitution on terms of the λ-calculus is defined by recursion on the structure of terms, as follows (note: x and y are only variables while M and N are any λ expression).

$x[x := \mathrm{N}] \equiv \mathrm{N}$ $y[x := \mathrm{N}] \equiv y$, if $x \neq y$ $(\mathrm{M}_1\ \mathrm{M}_2)[x := \mathrm{N}] \equiv (\mathrm{M}_1[x := \mathrm{N}])\ (\mathrm{M}_2[x := \mathrm{N}])$ $(\lambda x.\mathrm{M})[x := \mathrm{N}] \equiv \lambda x.\mathrm{M}$ $(\lambda y.\mathrm{M})[x := \mathrm{N}] \equiv \lambda y.(\mathrm{M}[x := \mathrm{N}])$, if $x \neq y$, *provided* $y \notin \mathrm{FV}(\mathrm{N})$

To substitute into a lambda abstraction, it is sometimes necessary to α-convert the expression. For example, it is not correct for (λ*x*.*y*)[*y* := *x*] to result in (λ*x*.*x*), because the substituted *x* was supposed to be free but ended up being bound. The correct substitution in this case is (λ*z*.*x*), up to α-equivalence. Notice that substitution is defined uniquely up to α-equivalence.

5.5.2 β-reduction

Beta-reduction captures the idea of function application. Beta-reduction is defined in terms of substitution: the beta-reduction of $((\lambda V.E)\ E')$ is $E[V := E']$.

For example, assuming some encoding of 2, 7, ×, we have the following β-reduction: ((λ*n*.*n*×2) 7) → 7×2.

5.5.3 η-conversion

Eta-conversion expresses the idea of extensionality, which in this context is that two functions are the same if and only if they give the same result for all arguments. Eta-conversion converts between $\lambda x.(f\ x)$ and f whenever x does not appear free in f.

5.6 Normal forms and confluence

Main article: Normalization property (abstract rewriting)

For the untyped lambda calculus, β-reduction as a rewriting rule is neither strongly normalising nor weakly normalising.

However, it can be shown that β-reduction is confluent. (Of course, we are working up to α-conversion, i.e. we consider two normal forms to be equal, if it is possible to α-convert one into the other.)

Therefore, both strongly normalising terms and weakly normalising terms have a unique normal form. For strongly normalising terms, any reduction strategy is guaranteed to yield the normal form, whereas for weakly normalising terms, some reduction strategies may fail to find it.

5.7 Encoding datatypes

Main articles: Church encoding and Mogensen–Scott encoding

The basic lambda calculus may be used to model booleans, arithmetic, data structures and recursion, as illustrated in the following sub-sections.

5.7.1 Arithmetic in lambda calculus

There are several possible ways to define the natural numbers in lambda calculus, but by far the most common are the Church numerals, which can be defined as follows:

$$0 := \lambda f.\lambda x.x$$
$$1 := \lambda f.\lambda x.f\ x$$
$$2 := \lambda f.\lambda x.f\ (f\ x)$$
$$3 := \lambda f.\lambda x.f\ (f\ (f\ x))$$

and so on. Or using the alternative syntax presented above in *Notation*:

$$0 := \lambda fx.x$$
$$1 := \lambda fx.f\ x$$
$$2 := \lambda fx.f\ (f\ x)$$
$$3 := \lambda fx.f\ (f\ (f\ x))$$

A Church numeral is a higher-order function—it takes a single-argument function f, and returns another single-argument function. The Church numeral n is a function that takes a function f as argument and returns the n-th composition of f, i.e. the function f composed with itself n times. This is denoted $f^{(n)}$ and is in fact the n-th power of f (considered as an operator); $f^{(0)}$ is defined to be the identity function. Such repeated compositions (of a single function f) obey the laws of exponents, which is why these numerals can be used for arithmetic. (In Church's original lambda calculus, the formal parameter of a lambda expression was required to occur at least once in the function body, which made the above definition of 0 impossible.)

We can define a successor function, which takes a number n and returns $n + 1$ by adding another application of f,where '(mf)x' means the function 'f' is applied 'm' times on 'x':

SUCC := $\lambda n.\lambda f.\lambda x.f\ (n\ f\ x)$

Because the m-th composition of f composed with the n-th composition of f gives the m+n-th composition of f, addition can be defined as follows:

PLUS := $\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$

PLUS can be thought of as a function taking two natural numbers as arguments and returning a natural number; it can be verified that

PLUS 2 3

and

5

are β-equivalent lambda expressions. Since adding m to a number n can be accomplished by adding 1 m times, an equivalent definition is:

PLUS := $\lambda m.\lambda n.m$ SUCC n [19]

Similarly, multiplication can be defined as

MULT := $\lambda m.\lambda n.\lambda f.m\ (n\ f)$[20]

Alternatively

MULT := $\lambda m.\lambda n.m$ (PLUS n) 0

since multiplying m and n is the same as repeating the add n function m times and then applying it to zero. Exponentiation has a rather simple rendering in Church numerals, namely

POW := $\lambda b.\lambda e.e\ b$

The predecessor function defined by PRED $n = n - 1$ for a positive integer n and PRED 0 = 0 is considerably more difficult. The formula

PRED := $\lambda n.\lambda f.\lambda x.n\ (\lambda g.\lambda h.h\ (g\ f))\ (\lambda u.x)\ (\lambda u.u)$

can be validated by showing inductively that if T denotes $(\lambda g.\lambda h.h\ (g\ f))$, then $T^{(n)}(\lambda u.x) = (\lambda h.h(f^{(n-1)}(x)))$ for $n > 0$. Two other definitions of PRED are given below, one using conditionals and the other using pairs. With the predecessor function, subtraction is straightforward. Defining

SUB := $\lambda m.\lambda n.n$ PRED m,

SUB m n yields $m - n$ when $m > n$ and 0 otherwise.

5.7.2 Logic and predicates

By convention, the following two definitions (known as Church booleans) are used for the boolean values TRUE and FALSE:

TRUE := $\lambda x.\lambda y.x$

FALSE := $\lambda x.\lambda y.y$

(Note that FALSE is equivalent to the Church numeral zero defined above)

Then, with these two λ-terms, we can define some logic operators (these are just possible formulations; other expressions are equally correct):

AND := $\lambda p.\lambda q.p\ q\ p$
OR := $\lambda p.\lambda q.p\ p\ q$
NOT := $\lambda p.$p FALSE TRUE
IFTHENELSE := $\lambda p.\lambda a.\lambda b.p\ a\ b$

We are now able to compute some logic functions, for example:

AND TRUE FALSE

≡ ($\lambda p.\lambda q.p\ q\ p$) TRUE FALSE $\rightarrow_\beta$ TRUE FALSE TRUE
≡ ($\lambda x.\lambda y.x$) FALSE TRUE $\rightarrow_\beta$ FALSE

and we see that AND TRUE FALSE is equivalent to FALSE.

A *predicate* is a function that returns a boolean value. The most fundamental predicate is ISZERO, which returns TRUE if its argument is the Church numeral 0, and FALSE if its argument is any other Church numeral:

ISZERO := $\lambda n.n$ ($\lambda x.$FALSE) TRUE

The following predicate tests whether the first argument is less-than-or-equal-to the second:

LEQ := $\lambda m.\lambda n.$ISZERO (SUB m n),

and since $m = n$, if LEQ m n and LEQ n m, it is straightforward to build a predicate for numerical equality.

The availability of predicates and the above definition of TRUE and FALSE make it convenient to write "if-then-else" expressions in lambda calculus. For example, the predecessor function can be defined as:

PRED := $\lambda n.n$ ($\lambda g.\lambda k.$ISZERO (g 1) k (PLUS (g k) 1)) ($\lambda v.$0) 0

which can be verified by showing inductively that n ($\lambda g.\lambda k.$ISZERO (g 1) k (PLUS (g k) 1)) ($\lambda v.$0) is the add $n - 1$ function for $n > 0$.

5.7.3 Pairs

A pair (2-tuple) can be defined in terms of TRUE and FALSE, by using the Church encoding for pairs. For example, PAIR encapsulates the pair (x,y), FIRST returns the first element of the pair, and SECOND returns the second.

PAIR := $\lambda x.\lambda y.\lambda f.f\ x\ y$
FIRST := $\lambda p.p$ TRUE
SECOND := $\lambda p.p$ FALSE
NIL := $\lambda x.$TRUE
NULL := $\lambda p.p$ ($\lambda x.\lambda y.$FALSE)

A linked list can be defined as either NIL for the empty list, or the PAIR of an element and a smaller list. The predicate NULL tests for the value NIL. (Alternatively, with NIL := FALSE, the construct l ($\lambda h.\lambda t.\lambda z.$deal_with_head_$h$_and_tail_$t$) (deal_with_nil) obviates the need for an explicit NULL test).

As an example of the use of pairs, the shift-and-increment function that maps (m, n) to (n, n + 1) can be defined as

Φ := $\lambda x.$PAIR (SECOND x) (SUCC (SECOND x))

which allows us to give perhaps the most transparent version of the predecessor function:

PRED := $\lambda n.$FIRST (n Φ (PAIR 0 0)).

5.7.4 Recursion and fixed points

Main article: Fixed-point combinator
See also: SKI combinator calculus § Self-application_and_recursion

Recursion is the definition of a function using the function itself. Lambda calculus cannot express this as directly as some other notations: all functions are anonymous in lambda calculus, so we can't refer to a value which is yet to be defined, inside the lambda term defining that same value.. However, recursion can still be achieved by arranging for a lambda expression to receive itself as its argument value, for example in $(\lambda x.x\ x)\ E$.

Consider the factorial function F(n) recursively defined by

F(n) = 1, if $n = 0$; else $n \times$ F($n - 1$).

In the lambda expression which is to represent this function, a *parameter* (typically the first one) will be assumed to receive the lambda expression itself as its value, so that calling it – applying it to an argument – will amount to recursion. Thus to achieve recursion, the intended-as-self-referencing argument (called r here) must always be passed to itself within the function body, at a call point:

G := $\lambda r.\ \lambda n.$(1, if $n = 0$; else $n \times (r\ r\ (n-1))$)

with $r\ r\ x$ = F x = G $r\ x$ to hold, so r = G and

F := G G = $(\lambda x.x\ x)$ G

The self-application achieves replication here, passing the function's lambda expression on to the next invocation as an argument value, making it available to be referenced and called there.

This solves it but requires re-writing each recursive call as self-application. We would like to have a generic solution, without a need for any re-writes:

G := $\lambda r.\ \lambda n.$(1, if $n = 0$; else $n \times (r\ (n-1))$)

with $r\ x$ = F x = G $r\ x$ to hold, so r = G r =: FIX G and

F := FIX G where FIX g := (r where $r = g\ r$) = g (FIX g)

so that FIX G = G (FIX G) = $(\lambda n.$(1, if $n = 0$; else $n \times$ ((FIX G) $(n-1)$)))

Given a lambda term with first argument representing recursive call (e.g. G here), the *fixed-point* combinator FIX will return a self-replicating lambda expression representing the recursive function (here, F). The function does not need to be explicitly passed to itself at any point, for the self-replication is arranged in advance, when it is created, to be done each time it is called. Thus the original lambda expression (FIX G) is re-created inside itself, at call-point, achieving self-reference.

In fact, there are many possible definitions for this FIX operator, the simplest of them being:

Y := $\lambda g.(\lambda x.g\ (x\ x))\ (\lambda x.g\ (x\ x))$

In the lambda calculus, **Y** g is a fixed-point of g, as it expands to:

Y g
$(\lambda h.(\lambda x.h\ (x\ x))\ (\lambda x.h\ (x\ x)))\ g$
$(\lambda x.g\ (x\ x))\ (\lambda x.g\ (x\ x))$
$g\ ((\lambda x.g\ (x\ x))\ (\lambda x.g\ (x\ x)))$
g (**Y** g)

Now, to perform our recursive call to the factorial function, we would simply call (**Y** G) n, where n is the number we are calculating the factorial of. Given $n = 4$, for example, this gives:

(**Y** G) 4
G (**Y** G) 4
($\lambda r.\lambda n.$(1, if $n = 0$; else $n \times (r\ (n{-}1))$)) (**Y** G) 4
($\lambda n.$(1, if $n = 0$; else $n \times$ ((**Y** G) $(n{-}1)$))) 4
1, if 4 = 0; else 4 × ((**Y** G) (4−1))
4 × (G (**Y** G) (4−1))
4 × (($\lambda n.$(1, if $n = 0$; else $n \times$ ((**Y** G) $(n{-}1)$))) (4−1))
4 × (1, if 3 = 0; else 3 × ((**Y** G) (3−1)))
4 × (3 × (G (**Y** G) (3−1)))
4 × (3 × (($\lambda n.$(1, if $n = 0$; else $n \times$ ((**Y** G) $(n{-}1)$))) (3−1)))
4 × (3 × (1, if 2 = 0; else 2 × ((**Y** G) (2−1))))
4 × (3 × (2 × (G (**Y** G) (2−1))))
4 × (3 × (2 × (($\lambda n.$(1, if $n = 0$; else $n \times$ ((**Y** G) $(n{-}1)$))) (2−1))))
4 × (3 × (2 × (1, if 1 = 0; else 1 × ((**Y** G) (1−1)))))
4 × (3 × (2 × (1 × (G (**Y** G) (1−1)))))
4 × (3 × (2 × (1 × (($\lambda n.$(1, if $n = 0$; else $n \times$ ((**Y** G) $(n{-}1)$))) (1−1)))))
4 × (3 × (2 × (1 × (1, if 0 = 0; else 0 × ((**Y** G) (0−1)))))))
4 × (3 × (2 × (1 × (1))))
24

Every recursively defined function can be seen as a fixed point of some suitably defined function closing over the recursive call with an extra argument, and therefore, using **Y**, every recursively defined function can be expressed as a lambda expression. In particular, we can now cleanly define the subtraction, multiplication and comparison predicate of natural numbers recursively.

5.7.5 Standard terms

Certain terms have commonly accepted names:

I := $\lambda x.x$
K := $\lambda x.\lambda y.x$
S := $\lambda x.\lambda y.\lambda z.x\ z\ (y\ z)$
B := $\lambda x.\lambda y.\lambda z.x\ (y\ z)$
C := $\lambda x.\lambda y.\lambda z.x\ z\ y$
W := $\lambda x.\lambda y.x\ y\ y$
U := $\lambda x.\lambda y.y\ (x\ x\ y)$
ω := $\lambda x.x\ x$
Ω := **ω ω**
Y := $\lambda g.(\lambda x.g\ (x\ x))\ (\lambda x.g\ (x\ x))$

5.8 Typed lambda calculus

Main article: Typed lambda calculus

A **typed lambda calculus** is a typed formalism that uses the lambda-symbol (λ) to denote anonymous function abstraction. In this context, types are usually objects of a syntactic nature that are assigned to lambda terms; the exact

nature of a type depends on the calculus considered (see kinds below). From a certain point of view, typed lambda calculi can be seen as refinements of the untyped lambda calculus but from another point of view, they can also be considered the more fundamental theory and *untyped lambda calculus* a special case with only one type.[21]

Typed lambda calculi are foundational programming languages and are the base of typed functional programming languages such as ML and Haskell and, more indirectly, typed imperative programming languages. Typed lambda calculi play an important role in the design of type systems for programming languages; here typability usually captures desirable properties of the program, e.g. the program will not cause a memory access violation.

Typed lambda calculi are closely related to mathematical logic and proof theory via the Curry–Howard isomorphism and they can be considered as the internal language of classes of categories, e.g. the simply typed lambda calculus is the language of Cartesian closed categories (CCCs).

5.9 Computable functions and lambda calculus

A function F: $\mathbf{N} \to \mathbf{N}$ of natural numbers is a computable function if and only if there exists a lambda expression f such that for every pair of x, y in $\mathbf{N}$, $F(x)=y$ if and only if $f\ x =_\beta y$, where x and y are the Church numerals corresponding to x and y, respectively and $=_\beta$ meaning equivalence with beta reduction. This is one of the many ways to define computability; see the Church-Turing thesis for a discussion of other approaches and their equivalence.

5.10 Undecidability of equivalence

There is no algorithm that takes as input two lambda expressions and outputs TRUE or FALSE depending on whether or not the two expressions are equivalent. This was historically the first problem for which undecidability could be proven. As is common for a proof of undecidability, the proof shows that no computable function can decide the equivalence. Church's thesis is then invoked to show that no algorithm can do so.

Church's proof first reduces the problem to determining whether a given lambda expression has a *normal form*. A normal form is an equivalent expression that cannot be reduced any further under the rules imposed by the form. Then he assumes that this predicate is computable, and can hence be expressed in lambda calculus. Building on earlier work by Kleene and constructing a Gödel numbering for lambda expressions, he constructs a lambda expression e that closely follows the proof of Gödel's first incompleteness theorem. If e is applied to its own Gödel number, a contradiction results.

5.11 Lambda calculus and programming languages

As pointed out by Peter Landin's 1965 paper A Correspondence between ALGOL 60 and Church's Lambda-notation, sequential procedural programming languages can be understood in terms of the lambda calculus, which provides the basic mechanisms for procedural abstraction and procedure (subprogram) application.

Lambda calculus reifies "functions" and makes them first-class objects, which raises implementation complexity when it is implemented.

5.11.1 Anonymous functions

Main article: Anonymous function

For example, in Lisp the 'square' function can be expressed as a lambda expression as follows:

```
(lambda (x) (* x x))
```

The above example is an expression that evaluates to a first-class function. The symbol lambda creates an anonymous function, given a list of parameter names, (x) — just a single argument in this case, and an expression that is evaluated as the body of the function, (* x x). The Haskell example is identical. Anonymous functions are sometimes called lambda expressions.

For example, Pascal and many other imperative languages have long supported passing subprograms as arguments to other subprograms through the mechanism of function pointers. However, function pointers are not a sufficient condition for functions to be first class datatypes, because a function is a first class datatype if and only if new instances of the function can be created at run-time. And this run-time creation of functions is supported in Smalltalk, Javascript, and more recently in Scala, Eiffel ("agents"), C# ("delegates") and C++11, among others.

5.11.2 Reduction strategies

For more details on this topic, see Evaluation strategy.

Whether a term is normalising or not, and how much work needs to be done in normalising it if it is, depends to a large extent on the reduction strategy used. The distinction between reduction strategies relates to the distinction in functional programming languages between eager evaluation and lazy evaluation.

Full beta reductions Any redex can be reduced at any time. This means essentially the lack of any particular reduction strategy—with regard to reducibility, "all bets are off".

Applicative order The rightmost, innermost redex is always reduced first. Intuitively this means a function's arguments are always reduced before the function itself. Applicative order always attempts to apply functions to normal forms, even when this is not possible.

Most programming languages (including Lisp, ML and imperative languages like C and Java) are described as "strict", meaning that functions applied to non-normalising arguments are non-normalising. This is done essentially using applicative order, call by value reduction (see below), but usually called "eager evaluation".

Normal order The leftmost, outermost redex is always reduced first. That is, whenever possible the arguments are substituted into the body of an abstraction before the arguments are reduced.

Call by name As normal order, but no reductions are performed inside abstractions. For example, $\lambda x.(\lambda x.x)x$ is in normal form according to this strategy, although it contains the redex $(\lambda x.x)x$.

Call by value Only the outermost redexes are reduced: a redex is reduced only when its right hand side has reduced to a value (variable or lambda abstraction).

Call by need As normal order, but function applications that would duplicate terms instead name the argument, which is then reduced only "when it is needed". Called in practical contexts "lazy evaluation". In implementations this "name" takes the form of a pointer, with the redex represented by a thunk.

Applicative order is not a normalising strategy. The usual counterexample is as follows: define $\mathbf{\Omega} = \omega\omega$ where $\boldsymbol{\omega} = \lambda x.xx$. This entire expression contains only one redex, namely the whole expression; its reduct is again $\mathbf{\Omega}$. Since this is the only available reduction, $\mathbf{\Omega}$ has no normal form (under any evaluation strategy). Using applicative order, the expression $\mathbf{KI\Omega} = (\lambda x.\lambda y.x)\ (\lambda x.x)\mathbf{\Omega}$ is reduced by first reducing $\mathbf{\Omega}$ to normal form (since it is the rightmost redex), but since $\mathbf{\Omega}$ has no normal form, applicative order fails to find a normal form for $\mathbf{KI\Omega}$.

In contrast, normal order is so called because it always finds a normalising reduction, if one exists. In the above example, $\mathbf{KI\Omega}$ reduces under normal order to I, a normal form. A drawback is that redexes in the arguments may be copied, resulting in duplicated computation (for example, $(\lambda x.xx)\ ((\lambda x.x)y)$ reduces to $((\lambda x.x)y)\ ((\lambda x.x)y)$ using this strategy; now there are two redexes, so full evaluation needs two more steps, but if the argument had been reduced first, there would now be none).

The positive tradeoff of using applicative order is that it does not cause unnecessary computation, if all arguments are used, because it never substitutes arguments containing redexes and hence never needs to copy them (which would duplicate work). In the above example, in applicative order $(\lambda x.xx)\ ((\lambda x.x)y)$ reduces first to $(\lambda x.xx)y$ and then to the normal order yy, taking two steps instead of three.

Most *purely* functional programming languages (notably Miranda and its descendents, including Haskell), and the proof languages of theorem provers, use *lazy evaluation*, which is essentially the same as call by need. This is like normal order reduction, but call by need manages to avoid the duplication of work inherent in normal order reduction using *sharing*. In the example given above, $(\lambda x.xx)\ ((\lambda x.x)y)$ reduces to $((\lambda x.x)y)\ ((\lambda x.x)y)$, which has two redexes, but in call by need they are represented using the same object rather than copied, so when one is reduced the other is too.

5.11.3 A note about complexity

While the idea of beta reduction seems simple enough, it is not an atomic step, in that it must have a non-trivial cost when estimating computational complexity.[22] To be precise, one must somehow find the location of all of the occurrences of the bound variable V in the expression E, implying a time cost, or one must keep track of these locations in some way, implying a space cost. A naïve search for the locations of V in E is $O(n)$ in the length n of E. This has led to the study of systems that use explicit substitution. Sinot's director strings[23] offer a way of tracking the locations of free variables in expressions.

5.11.4 Parallelism and concurrency

The Church–Rosser property of the lambda calculus means that evaluation (β-reduction) can be carried out in *any order*, even in parallel. This means that various nondeterministic evaluation strategies are relevant. However, the lambda calculus does not offer any explicit constructs for parallelism. One can add constructs such as Futures to the lambda calculus. Other process calculi have been developed for describing communication and concurrency.

5.12 Semantics

The fact that lambda calculus terms act as functions on other lambda calculus terms, and even on themselves, led to questions about the semantics of the lambda calculus. Could a sensible meaning be assigned to lambda calculus terms? The natural semantics was to find a set D isomorphic to the function space $D \to D$, of functions on itself. However, no nontrivial such D can exist, by cardinality constraints because the set of all functions from D to D has greater cardinality than D, unless D is a singleton set.

In the 1970s, Dana Scott showed that, if only continuous functions were considered, a set or domain D with the required property could be found, thus providing a model for the lambda calculus.

This work also formed the basis for the denotational semantics of programming languages.

5.13 See also

- Applicative computing systems – Treatment of objects in the style of the lambda calculus
- Binary lambda calculus – A version of lambda calculus with binary I/O, a binary encoding of terms, and a designated universal machine.
- Calculus of constructions – A typed lambda calculus with types as first-class values
- Cartesian closed category – A setting for lambda calculus in category theory
- Categorical abstract machine – A model of computation applicable to lambda calculus
- Combinatory logic – A notation for mathematical logic without variables
- Curry-Howard isomorphism – The formal correspondence between programs and proofs
- Deductive lambda calculus – The consideration of the problems associated with considering lambda calculus as a Deductive system.
- Domain theory – Study of certain posets giving denotational semantics for lambda calculus
- Evaluation strategy – Rules for the evaluation of expressions in programming languages
- Explicit substitution – The theory of substitution, as used in β-reduction
- Functional programming
- Harrop formula – A kind of constructive logical formula such that proofs are lambda terms
- Kappa calculus – A first-order analogue of lambda calculus

- Kleene-Rosser paradox – A demonstration that some form of lambda calculus is inconsistent
- Knights of the Lambda Calculus – A semi-fictional organization of LISP and Scheme hackers
- Lambda calculus definition - Formal definition of the lambda calculus.
- Lambda cube – A framework for some extensions of typed lambda calculus
- Lambda-mu calculus – An extension of the lambda calculus for treating classical logic
- Let expression – An expression close related to a lambda abstraction.
- Minimalism (computing)
- Rewriting – Transformation of formulæ in formal systems
- SECD machine – A virtual machine designed for the lambda calculus
- Simply typed lambda calculus - Version(s) with a single type constructor
- SKI combinator calculus – A computational system based on the **S**, **K** and **I** combinators
- System F – A typed lambda calculus with type-variables
- Typed lambda calculus – Lambda calculus with typed variables (and functions)
- Universal Turing machine – A formal computing machine that is equivalent to lambda calculus
- Unlambda – An esoteric functional programming language based on combinatory logic

5.14 Further reading

- Abelson, Harold & Gerald Jay Sussman. Structure and Interpretation of Computer Programs. The MIT Press. ISBN 0-262-51087-1.
- Hendrik Pieter Barendregt *Introduction to Lambda Calculus*.
- Henk Barendregt, The Impact of the Lambda Calculus in Logic and Computer Science. The Bulletin of Symbolic Logic, Volume 3, Number 2, June 1997.
- Barendregt, Hendrik Pieter, *The Type Free Lambda Calculus* pp1091–1132 of *Handbook of Mathematical Logic*, North-Holland (1977) ISBN 0-7204-2285-X
- Cardone and Hindley, 2006. History of Lambda-calculus and Combinatory Logic. In Gabbay and Woods (eds.), *Handbook of the History of Logic*, vol. 5. Elsevier.
- Church, Alonzo, *An unsolvable problem of elementary number theory*, American Journal of Mathematics, 58 (1936), pp. 345–363. This paper contains the proof that the equivalence of lambda expressions is in general not decidable.
- Alonzo Church, *The Calculi of Lambda-Conversion* (ISBN 978-0-691-08394-0)[24]
- Kleene, Stephen, *A theory of positive integers in formal logic*, American Journal of Mathematics, 57 (1935), pp. 153–173 and 219–244. Contains the lambda calculus definitions of several familiar functions.
- Landin, Peter, *A Correspondence Between ALGOL 60 and Church's Lambda-Notation*, Communications of the ACM, vol. 8, no. 2 (1965), pages 89–101. Available from the ACM site. A classic paper highlighting the importance of lambda calculus as a basis for programming languages.
- Larson, Jim, *An Introduction to Lambda Calculus and Scheme*. A gentle introduction for programmers.
- Schalk, A. and Simmons, H. (2005) *An introduction to λ-calculi and arithmetic with a decent selection of exercises. Notes for a course in the Mathematical Logic MSc at Manchester University.*

- de Queiroz, Ruy J.G.B. (2008) *On Reduction Rules, Meaning-as-Use and Proof-Theoretic Semantics*. Studia Logica, 90(2):211-247. A paper giving a formal underpinning to the idea of 'meaning-is-use' which, even if based on proofs, it is different from proof-theoretic semantics as in the Dummett–Prawitz tradition since it takes reduction as the rules giving meaning.

Monographs/textbooks for graduate students:

- Morten Heine Sørensen, Paweł Urzyczyn, *Lectures on the Curry-Howard isomorphism*, Elsevier, 2006, ISBN 0-444-52077-5 is a recent monograph that covers the main topics of lambda calculus from the type-free variety, to most typed lambda calculi, including more recent developments like pure type systems and the lambda cube. It does not cover subtyping extensions.
- Pierce, Benjamin (2002), *Types and Programming Languages*, MIT Press, ISBN 0-262-16209-1 covers lambda calculi from a practical type system perspective; some topics like dependent types are only mentioned, but subtyping is an important topic.

Some parts of this article are based on material from FOLDOC, used with permission.

5.15 External links

- Hazewinkel, Michiel, ed. (2001), "Lambda-calculus", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Achim Jung, *A Short Introduction to the Lambda Calculus*-(PDF)
- Dana Scott, *A timeline of lambda calculus*-(PDF)
- David C. Keenan, *To Dissect a Mockingbird: A Graphical Notation for the Lambda Calculus with Animated Reduction*
- Raúl Rojas, *A Tutorial Introduction to the Lambda Calculus*-(PDF)
- Peter Selinger, *Lecture Notes on the Lambda Calculus*-(PDF)
- L. Allison, *Some executable λ-calculus examples*
- Georg P. Loczewski, *The Lambda Calculus and A++*
- Bret Victor, *Alligator Eggs: A Puzzle Game Based on Lambda Calculus*
- *Lambda Calculus* on Safalra's Website
- *Lambda Calculus at PlanetMath.org.*
- LCI Lambda Interpreter a simple yet powerful pure calculus interpreter
- Lambda Calculus links on Lambda-the-Ultimate
- Mike Thyer, Lambda Animator, a graphical Java applet demonstrating alternative reduction strategies.
- Implementing the Lambda calculus using C++ Templates
- Marius Buliga, *Graphic lambda calculus*
- *Lambda Calculus as a Workflow Model* by Peter Kelly, Paul Coddington, and Andrew Wendelborn; mentions graph reduction as a common means of evaluating lambda expressions and discusses the applicability of lambda calculus for distributed computing (due to the Church–Rosser property, which enables parallel graph reduction for lambda expressions).
- Shane Steinert-Threlkeld, "Lambda Calculi", *Internet Encyclopedia of Philosophy*

5.16 References

[1] Turing, A. M. (December 1937). "Computability and λ-Definability". *The Journal of Symbolic Logic* **2** (4): 153–163. doi:10.2307/2268280. JSTOR 2268280.

[2] Coquand, Thierry, "Type Theory", *The Stanford Encyclopedia of Philosophy* (Summer 2013 Edition), Edward N. Zalta (ed.).

[3] *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus - Michael Moortgat - Google Books*, Books.google.co.uk, 1988-01-01, ISBN 9789067653879, retrieved 2013-09-15

[4] *Computing Meaning - Google Books*, Books.google.co.uk, 2008-07-02, ISBN 9781402059575, retrieved 2013-09-15

[5] Mitchell, John C. (2003), *Concepts in Programming Languages*, Cambridge University Press, p. 57, ISBN 9780521780988.

[6] Basic Category Theory for Computer Scientists, p. 53, Benjamin C. Pierce

[7] A. Church, "A set of postulates for the foundation of logic", *Annals of Mathematics*, Series 2, 33:346–366 (1932).

[8] For a full history, see Cardone and Hindley's "History of Lambda-calculus and Combinatory Logic" (2006).

[9] A. Church, "An unsolvable problem of elementary number theory", *American Journal of Mathematics*, Volume 58, No. 2. (April 1936), pp. 345-363.

[10] Church, A. "A Formulation of the Simple Theory of Types". *Journal of Symbolic Logic* **5**: 1940. doi:10.2307/2266170.

[11] Alama, Jesse "The Lambda Calculus", *The Stanford Encyclopedia of Philosophy* (Summer 2013 Edition), Edward N. Zalta (ed.).

[12] Barendregt, Hendrik Pieter (1984), *The Lambda Calculus: Its Syntax and Semantics*, Studies in Logic and the Foundations of Mathematics **103** (Revised ed.), North Holland, Amsterdam. Corrections, ISBN 0-444-87508-5 External link in |publisher= (help)

[13] "Example for Rules of Associativity". Lambda-bound.com. Retrieved 2012-06-18.

[14] Selinger, Peter (2008), *Lecture Notes on the Lambda Calculus* (PDF) **0804** (class: cs.LO), Department of Mathematics and Statistics, University of Ottawa, p. 9, arXiv:0804.3434, Bibcode:2008arXiv0804.3434S

[15] "Example for Rule of Associativity". Lambda-bound.com. Retrieved 2012-06-18.

[16] Barendregt, Henk; Barendsen, Erik (March 2000), *Introduction to Lambda Calculus* (PDF)

[17] de Queiroz, Ruy J.G.B. "A Proof-Theoretic Account of Programming and the Role of Reduction Rules." *Dialectica* **42**(4), pages 265-282, 1988.

[18] Turbak, Franklyn; Gifford, David (2008), *Design concepts in programming languages*, MIT press, p. 251, ISBN 978-0-262-20175-9

[19] Felleisen, Matthias; Flatt, Matthew (2006), *Programming Languages and Lambda Calculi* (PDF), p. 26

[20] Selinger, Peter (2008), *Lecture Notes on the Lambda Calculus* (PDF) **0804** (class: cs.LO), Department of Mathematics and Statistics, University of Ottawa, p. 16, arXiv:0804.3434, Bibcode:2008arXiv0804.3434S

[21] Types and Programming Languages, p. 273, Benjamin C. Pierce

[22] R. Statman, "The typed λ-calculus is not elementary recursive." *Theoretical Computer Science*, (1979) **9** pp73-81.

[23] F.-R. Sinot. "Director Strings Revisited: A Generic Approach to the Efficient Representation of Free Variables in Higher-order Rewriting." *Journal of Logic and Computation* **15**(2), pages 201-218, 2005.

[24] Frink Jr., Orrin (1944). "Review: *The Calculi of Lambda-Conversion* by Alonzo Church" (PDF). *Bull. Amer. Math. Soc.* **50** (3): 169–172. doi:10.1090/s0002-9904-1944-08090-7.

Chapter 6

Gödel's incompleteness theorems

Gödel's incompleteness theorems are two theorems of mathematical logic that demonstrate the inherent limitations of every formal axiomatic system containing basic arithmetic.[1] These results, published by Kurt Gödel in 1931, are important both in mathematical logic and in the philosophy of mathematics. The theorems are widely, but not universally, interpreted as showing that Hilbert's program to find a complete and consistent set of axioms for all mathematics is impossible.

The first incompleteness theorem states that no consistent system of axioms whose theorems can be listed by an effective procedure (i.e., an algorithm) is capable of proving all truths about the arithmetic of the natural numbers. For any such formal system, there will always be statements about the natural numbers that are true, but that are unprovable within the system. The second incompleteness theorem, an extension of the first, shows that the system can not demonstrate its own consistency.

Gödel's incompleteness theorems were the first of several closely related theorems on the limitations of formal systems. They were followed by Tarski's undefinability theorem on the formal undefinability of truth, Church's proof that Hilbert's Entscheidungsproblem is unsolvable, and Turing's theorem that there is no algorithm to solve the halting problem.

6.1 Formal systems: completeness, consistency, and effective axiomatization

The incompleteness theorems apply to formal systems that are of sufficient complexity to express the basic arithmetic of the natural numbers and which are complete, consistent, and effectively axiomatized, these concepts being detailed below. Particularly in the context of first-order logic, formal systems are also called *formal theories*. In general, a formal system is a deductive apparatus that consists of a particular set of axioms along with rules of symbolic manipulation (or rules of inference) that allow for the derivation of new theorems from the axioms. One example of such a system is first-order Peano arithmetic, a system in which all variables are intended to denote natural numbers. In other systems, such as set theory, only some sentences of the formal system express statements about the natural numbers. The incompleteness theorems are about formal provability within these systems, rather than about "provability" in an informal sense.

There are several properties that a formal system may have, including completeness, consistency, and the existence of an effective axiomatization. The incompleteness theorems show that systems which contain a sufficient amount of arithmetic cannot possess all three of these properties.

6.1.1 Effective axiomatization

A formal system is said to be *effectively axiomatized* (also called *effectively generated*) if its set of theorems is a recursively enumerable set (Franzén 2004, p. 112).

This means that there is a computer program that, in principle, could enumerate all the theorems of the system without listing any statements that are not theorems. Examples of effectively generated theories include Peano arithmetic and Zermelo–Fraenkel set theory.

The theory known as true arithmetic consists of all true statements about the standard integers in the language of Peano arithmetic. This theory is consistent, and complete, and contains a sufficient amount of arithmetic. However it does not have a recursively enumerable set of axioms, and thus does not satisfy the hypotheses of the incompleteness theorems.

6.1.2 Completeness

A set of axioms is complete if, for any statement in the axioms' language, that statement or its negation is provable from the axioms (Smith 2007, p. 24).

A formal system might be incomplete simply because not all the necessary axioms have been discovered or included. For example, Euclidean geometry without the parallel postulate is incomplete, because it is not possible to prove or disprove the parallel postulate from the remaining axioms. Similarly, the theory of dense linear orders is not complete, but becomes complete with an extra axiom stating that there are no endpoints in the order. The continuum hypothesis is a statement in the language of ZFC that is not provable within ZFC, so ZFC is not complete. In this case, there is no obvious candidate for a new axiom that resolves the issue.

The theory of first-order Peano arithmetic is consistent, has an infinite but recursively enumerable set of axioms, and can encode enough arithmetic for the hypotheses of the incompleteness theorem. Thus, by the first incompleteness theorem, Peano Arithmetic is not complete. The theorem gives an explicit example of a statement of arithmetic that is true (in the usual model) but not provable in Peano arithmetic. Moreover, no effectively axiomatized, consistent extension of Peano arithmetic can be complete.

6.1.3 Consistency

A set of axioms is (simply) consistent if there is no statement such that both the statement and its negation are provable from the axioms, and *inconsistent* otherwise.

Peano arithmetic is provably consistent from ZFC, but not from within itself. Similarly, ZFC is not provably consistent from within itself, but ZFC + "there exists an inaccessible cardinal" proves ZFC is consistent because if κ is the least such cardinal, then $V\kappa$ sitting inside the von Neumann universe is a model of ZFC, and a theory is consistent if and only if it has a model.

If one takes all statements in the language of Peano arithmetic as axioms, then this theory is complete, has a recursively enumerable set of axioms, and can describe addition and multiplication. However, it is not consistent.

Additional examples of inconsistent theories arise from the paradoxes that result when the axiom schema of unrestricted comprehension is assumed in set theory.

6.1.4 Systems which contain arithmetic

The incompleteness theorems apply only to formal systems which are able to prove a sufficient collection of facts about the natural numbers. One sufficient collection is the set of theorems of Robinson arithmetic Q. Some systems, such as Peano arithmetic, can directly express statements about natural numbers. Others, such as ZFC set theory, are able to interpret statements about natural numbers into their language. Either of these options is appropriate for the incompleteness theorems.

The theory of algebraically closed fields of a given characteristic is complete, consistent, and has an infinite but recursively enumerable set of axioms. However it is not possible to encode the integers into this theory, and the theory cannot describe arithmetic of integers. A similar example is the theory of real closed fields, which is essentially equivalent to Tarski's axioms for Euclidean geometry. So Euclidean geometry itself (in Tarski's formulation) is an example of a complete, consistent, effectively axiomatized theory.

The system of Presburger arithmetic consists of a set of axioms for the natural numbers with just the addition operation (multiplication is omitted). Presburger arithmetic is complete, consistent, and recursively enumerable and can encode addition but not multiplication of natural numbers, showing that for Gödel's theorems one needs the theory to encode not just addition but also multiplication.

Dan Willard (2001) has studied some weak families of arithmetic systems which allow enough arithmetic as relations to formalise Gödel numbering, but which are not strong enough to have multiplication as a function, and so fail to

prove the second incompleteness theorem; these systems are consistent and capable of proving their own consistency (see self-verifying theories).

6.1.5 Conflicting goals

In choosing a set of axioms, one goal is to be able to prove as many correct results as possible, without proving any incorrect results. For example, we could imagine a set of true axioms which allow us to prove every true arithmetical claim about the natural numbers (Smith 2007, p 2). In the standard system of first-order logic, an inconsistent set of axioms will prove every statement in its language (this is sometimes called the principle of explosion), and is thus automatically complete. A set of axioms that is both complete and consistent, however, proves a maximal set of non-contradictory theorems (Hinman 2005, p. 143).

The pattern illustrated in the previous sections with Peano arithmetic, ZFC, and ZFC + "there exists an inaccessible cardinal" cannot generally be broken. Here ZFC + "there exists an inaccessible cardinal" cannot from itself, be proved consistent. It is also not complete, as illustrated by the in ZFC + "there exists an inaccessible cardinal" theory unresolved continuum hypothesis.

The first incompleteness theorem shows that, in formal systems that can express basic arithmetic, a complete and consistent finite list of axioms can never be created: each time an additional, consistent statement is added as an axiom, there are other true statements that still cannot be proved, even with the new axiom. If an axiom is ever added that makes the system complete, it does so at the cost of making the system inconsistent. It is not even possible for an infinite list of axioms to be complete, consistent, and effectively axiomatized.

6.2 First incompleteness theorem

Gödel's first incompleteness theorem first appeared as "Theorem VI" in Gödel's 1931 paper *On Formally Undecidable Propositions of Principia Mathematica and Related Systems I*.[2] The hypotheses of the theorem were improved shortly thereafter by J. Barkley Rosser (1936) using Rosser's trick.

The resulting theorem (incorporating Rosser's improvement) may be paraphrased in English as follows, where "formal system" includes the assumption that the system is effectively generated.

> **First Incompleteness Theorem**: "Any consistent formal system F within which a certain amount of elementary arithmetic can be carried out is incomplete; i.e., there are statements of the language of F which can neither be proved nor disproved in F." (Raatikainen 2015)

The unprovable statement GF referred to by the theorem is often referred to as "the Gödel sentence" for the system F. The proof constructs a particular Gödel sentence for the system F, but there are infinitely many statements in the language of the system that share the same properties, such as the conjunction of the Gödel sentence and any logically valid sentence.

Each effectively generated system has its own Gödel sentence. It is possible to define a larger system F' that contains the whole of F plus GF as an additional axiom. This will not result in a complete system, because Gödel's theorem will also apply to F', and thus F' also cannot be complete. In this case, GF is indeed a theorem in F', because it is an axiom. Because GF states only that it is not provable in F, no contradiction is presented by its provability within F'. However, because the incompleteness theorem applies to F', there will be a new Gödel statement GF' for F', showing that F' is also incomplete. GF'' will differ from GF in that GF' will refer to F', rather than F.

6.2.1 Syntactic form of the Gödel sentence

The Gödel sentence is designed to refer, indirectly, to itself. The sentence states that, when a particular sequence of steps is used to construct another sentence, that constructed sentence will not be provable in F. However, the sequence of steps is such that the constructed sentence turns out to be GF itself. In this way, the Gödel sentence GF indirectly states its own unprovability within F (Smith 2007, p. 135).

To prove the first incompleteness theorem, Gödel demonstrated that the notion of provability within a system could be expressed purely in terms of arithmetical functions that operate on Gödel numbers of sentences of the system. Therefore, the system, which can prove certain facts about numbers, can also indirectly prove facts about its own

statements, provided that it is effectively generated. Questions about the provability of statements within the system are represented as questions about the arithmetical properties of numbers themselves, which would be decidable by the system if it were complete.

Thus, although the Gödel sentence refers indirectly to sentences of the system *F*, the Gödel sentence is actually written as a statement about natural numbers solely. It asserts that no natural number has a particular property, where that property is given by a primitive recursive relation (Smith 2007, p. 141). As such, the Gödel sentence can be written in the language of arithmetic with a simple syntactic form. In particular, it can be expressed as formula consisting of a number of leading universal quantifiers followed by a quantifier-free body (these formulas are at level Π_1^0 of the arithmetical hierarchy). Via the MRDP theorem, the Gödel sentence can be re-written as a statement that a particular polynomial in many variables with integer coefficients never takes the value zero when integers are substituted for its variables (Franzén 2005, p. 71).

6.2.2 Truth of the Gödel sentence

The first incompleteness theorem shows that the Gödel sentence *GF* of an appropriate formal theory *F* is unprovable in *F*. Because this unprovability is exactly what the sentence (indirectly) asserts, the Gödel sentence is, in fact, true (Smoryński 1977 p. 825; also see Franzén 2004 pp. 28–33). For this reason, the sentence *GF* is often said to be "true but unprovable." (Raatikainen 2015). The truth of the sentence *GF* may only be arrived at via a meta-analysis from outside the system. In general, this meta-analysis can be carried out within the weak formal system known as primitive recursive arithmetic, which proves the implication Con(*F*)→*GF*, where Con(*F*) is a canonical sentence asserting the consistency of *F* (Smoryński 1977 p. 840, Kikuchi and Tanaka 1994 p. 403).

Although the Gödel sentence of a consistent theory is true as a statement about the intended interpretation of arithmetic, the Gödel sentence will be false in some nonstandard models of arithmetic, as a consequence of Gödel's completeness theorem (Franzén 2005, p. 135). That theorem shows that, when a sentence is independent of a theory, the theory will have models in which the sentence is true and models in which the sentence is false. As described earlier, the Gödel sentence of a system *F* is an arithmetical statement which claims that no number exists with a particular property. The incompleteness theorem shows that this claim will be independent of the system *F*, and the truth of the Gödel sentence follows from the fact that no standard natural number has the property in question. Any model of in which the Gödel sentence is false must contain some element which satisfies the property within that model. Such a model must be "nonstandard" – it must contain elements that do not correspond to any standard natural number (Raatikainen 2015, Franzén 2005, p. 135).

6.2.3 Relationship with the liar paradox

Gödel specifically cites Richard's paradox and the liar paradox as semantical analogues to his syntactical incompleteness result in the introductory section of On Formally Undecidable Propositions in Principia Mathematica and Related Systems I. The liar paradox is the sentence "This sentence is false." An analysis of the liar sentence shows that it cannot be true (for then, as it asserts, it is false), nor can it be false (for then, it is true). A Gödel sentence *G* for a system *F* makes a similar assertion to the liar sentence, but with truth replaced by provability: *G* says "*G* is not provable in the system *F*." The analysis of the truth and provability of *G* is a formalized version of the analysis of the truth of the liar sentence.

It is not possible to replace "not provable" with "false" in a Gödel sentence because the predicate "Q is the Gödel number of a false formula" cannot be represented as a formula of arithmetic. This result, known as Tarski's undefinability theorem, was discovered independently by both Gödel, when he was working on the proof of the incompleteness theorem, and by the theorem's namesake, Alfred Tarski.

6.2.4 Extensions of Gödel's original result

Compared to the theorems stated in Gödel's 1931 paper, many contemporary statements of the incompleteness theorems are more general in two ways. These generalized statements are phrased to apply to a broader class of systems, and they are phrased to incorporate weaker consistency assumptions.

Gödel demonstrated the incompleteness of the system of *Principia Mathematica*, a particular system of arithmetic, but a parallel demonstration could be given for any effective system of a certain expressiveness. Gödel commented on this fact in the introduction to his paper, but restricted the proof to one system for concreteness. In modern

statements of the theorem, it is common to state the effectiveness and expressiveness conditions as hypotheses for the incompleteness theorem, so that it is not limited to any particular formal system. The terminology used to state these conditions was not yet developed in 1931 when Gödel published his results.

Gödel's original statement and proof of the incompleteness theorem requires the assumption that the system is not just consistent but *ω-consistent*. A system is **ω-consistent** if it is not ω-inconsistent, and is ω-inconsistent if there is a predicate P such that for every specific natural number m the system proves ~$P(m)$, and yet the system also proves that there exists a natural number n such that $P(n)$. That is, the system says that a number with property P exists while denying that it has any specific value. The ω-consistency of a system implies its consistency, but consistency does not imply ω-consistency. J. Barkley Rosser (1936) strengthened the incompleteness theorem by finding a variation of the proof (Rosser's trick) that only requires the system to be consistent, rather than ω-consistent. This is mostly of technical interest, because all true formal theories of arithmetic (theories whose axioms are all true statements about natural numbers) are ω-consistent, and thus Gödel's theorem as originally stated applies to them. The stronger version of the incompleteness theorem that only assumes consistency, rather than ω-consistency, is now commonly known as Gödel's incompleteness theorem and as the Gödel–Rosser theorem.

6.3 Second incompleteness theorem

For each formal system F containing basic arithmetic, it is possible to canonically define a formula Cons(F) expressing the consistency of F. This formula expresses the property that "there does not exist a natural number coding a formal derivation within the system F whose conclusion is a syntactic contradiction." The syntactic contradiction is often taken to be "0=1", in which case Cons(F) states "there is no natural number that codes a derivation of '0=1' from the axioms of F."

Gödel's second incompleteness theorem shows that, under general assumptions, this canonical consistency statement Cons(F) will not be provable in F. The theorem first appeared as "Theorem XI" in Gödel's 1931 paper *On Formally Undecidable Propositions in Principia Mathematica and Related Systems I*. In the following statement, the term "formalized system" also includes an assumption that F is effectively axiomatized.

> **Second Incompleteness Theorem**: "Assume F is a consistent formalized system which contains elementary arithmetic. Then $F \nvdash \text{Cons}(F)$." (Raatikainen 2015)

This theorem is stronger than the first incompleteness theorem because the statement constructed in the first incompleteness theorem does not directly express the consistency of the system. The proof of the second incompleteness theorem is obtained by formalizing the proof of the first incompleteness theorem within the system F itself.

6.3.1 Expressing consistency

There is a technical subtlety in the second incompleteness theorem regarding the method of expressing the consistency of F as a formula in the language of F. There are many ways to express the consistency of a system, and not all of them lead to the same result. The formula Cons(F) from the second incompleteness theorem is a particular expression of consistency.

Other formalizations of the claim that F is consistent may be inequivalent in F, and some may even be provable. For example, first-order Peano arithmetic (PA) can prove that "the largest consistent subset of PA" is consistent. But, because PA is consistent, the largest consistent subset of PA is just PA, so in this sense PA "proves that it is consistent". What PA does not prove is that the largest consistent subset of PA is, in fact, the whole of PA. (The term "largest consistent subset of PA" is meant here to be the largest consistent initial segment of the axioms of PA under some particular effective enumeration).

6.3.2 The Hilbert–Bernays conditions

The standard proof of the second incompleteness theorem assumes that the provability predicate ProvA(P) satisfies the Hilbert–Bernays provability conditions. Letting #(P) represent the Gödel number of a formula P, the derivability conditions say:

1. If F proves P, then F proves ProvA(#(P)).

2. F proves 1.; that is, F proves that if F proves P, then F proves ProvA(#(P)). In other words, F proves that ProvA(#(P)) implies ProvA(#(ProvA(#(P)))).

3. F proves that if F proves that ($P \rightarrow Q$) and F proves P then F proves Q. In other words, F proves that ProvA(#($P \rightarrow Q$)) and ProvA(#(P)) imply ProvA(#(Q)).

There are systems, such as Robinson arithmetic, which are strong enough to meet the assumptions of the first incompleteness theorem, but which do not prove the Hilbert–Bernays conditions. Peano arithmetic, however, is strong enough to verify these conditions, as are all theories stronger than Peano arithmetic.

6.3.3 Implications for consistency proofs

Gödel's second incompleteness theorem also implies that a system F_1 satisfying the technical conditions outlined above cannot prove the consistency of any system F_2 that proves the consistency of F_1. This is because such a system T_1 can prove that if F_2 proves the consistency of F_1, then F_1 is in fact consistent. For the claim that F_1 is consistent has form "for all numbers n, n has the decidable property of not being a code for a proof of contradiction in F_1". If F_1 were in fact inconsistent, then F_2 would prove for some n that n is the code of a contradiction in F_1. But if F_2 also proved that F_1 is consistent (that is, that there is no such n), then it would itself be inconsistent. This reasoning can be formalized in F_1 to show that if F_2 is consistent, then F_1 is consistent. Since, by second incompleteness theorem, F_1 does not prove its consistency, it cannot prove the consistency of F_2 either.

This corollary of the second incompleteness theorem shows that there is no hope of proving, for example, the consistency of Peano arithmetic using any finitistic means that can be formalized in a system the consistency of which is provable in Peano arithmetic. For example, the system of primitive recursive arithmetic (PRA), which is widely accepted as an accurate formalization of finitistic mathematics, is provably consistent in PA. Thus PRA cannot prove the consistency of PA. This fact is generally seen to imply that Hilbert's program, which aimed to justify the use of "ideal" (infinitistic) mathematical principles in the proofs of "real" (finitistic) mathematical statements by giving a finitistic proof that the ideal principles are consistent, cannot be carried out (Franzén 2004, p. 106).

The corollary also indicates the epistemological relevance of the second incompleteness theorem. It would actually provide no interesting information if a system F proved its consistency. This is because inconsistent theories prove everything, including their consistency. Thus a consistency proof of F in F would give us no clue as to whether F really is consistent; no doubts about the consistency of F would be resolved by such a consistency proof. The interest in consistency proofs lies in the possibility of proving the consistency of a system F in some system F' that is in some sense less doubtful than F itself, for example weaker than F. For many naturally occurring theories F and F', such as F = Zermelo–Fraenkel set theory and F' = primitive recursive arithmetic, the consistency of F' is provable in F, and thus F' cannot prove the consistency of F by the above corollary of the second incompleteness theorem.

The second incompleteness theorem does not rule out consistency proofs altogether, only consistency proofs that can be formalized in the system that is proved consistent. For example, Gerhard Gentzen proved the consistency of Peano arithmetic (PA) in a different system that includes an axiom asserting that the ordinal called ε_0 is wellfounded; see Gentzen's consistency proof. Gentzen's theorem spurred the development of ordinal analysis in proof theory.

6.4 Examples of undecidable statements

See also: List of statements independent of ZFC

There are two distinct senses of the word "undecidable" in mathematics and computer science. The first of these is the proof-theoretic sense used in relation to Gödel's theorems, that of a statement being neither provable nor refutable in a specified deductive system. The second sense, which will not be discussed here, is used in relation to computability theory and applies not to statements but to decision problems, which are countably infinite sets of questions each requiring a yes or no answer. Such a problem is said to be undecidable if there is no computable function that correctly answers every question in the problem set (see undecidable problem).

Because of the two meanings of the word undecidable, the term independent is sometimes used instead of undecidable for the "neither provable nor refutable" sense.

Undecidability of a statement in a particular deductive system does not, in and of itself, address the question of whether the truth value of the statement is well-defined, or whether it can be determined by other means. Undecidability only

implies that the particular deductive system being considered does not prove the truth or falsity of the statement. Whether there exist so-called "absolutely undecidable" statements, whose truth value can never be known or is ill-specified, is a controversial point in the philosophy of mathematics.

The combined work of Gödel and Paul Cohen has given two concrete examples of undecidable statements (in the first sense of the term): The continuum hypothesis can neither be proved nor refuted in ZFC (the standard axiomatization of set theory), and the axiom of choice can neither be proved nor refuted in ZF (which is all the ZFC axioms *except* the axiom of choice). These results do not require the incompleteness theorem. Gödel proved in 1940 that neither of these statements could be disproved in ZF or ZFC set theory. In the 1960s, Cohen proved that neither is provable from ZF, and the continuum hypothesis cannot be proved from ZFC.

In 1973, Saharon Shelah showed that the Whitehead problem in group theory is undecidable, in the first sense of the term, in standard set theory.

Gregory Chaitin produced undecidable statements in algorithmic information theory and proved another incompleteness theorem in that setting. Chaitin's incompleteness theorem states that for any system that can represent enough arithmetic, there is an upper bound c such that no specific number can be proved in that system to have Kolmogorov complexity greater than c. While Gödel's theorem is related to the liar paradox, Chaitin's result is related to Berry's paradox.

6.4.1 Undecidable statements provable in larger systems

These are natural mathematical equivalents of the Gödel "true but undecidable" sentence. They can be proved in a larger system which is generally accepted as a valid form of reasoning, but are undecidable in a more limited system such as Peano Arithmetic.

In 1977, Paris and Harrington proved that the Paris–Harrington principle, a version of the infinite Ramsey theorem, is undecidable in (first-order) Peano arithmetic, but can be proved in the stronger system of second-order arithmetic. Kirby and Paris later showed that Goodstein's theorem, a statement about sequences of natural numbers somewhat simpler than the Paris–Harrington principle, is also undecidable in Peano arithmetic.

Kruskal's tree theorem, which has applications in computer science, is also undecidable from Peano arithmetic but provable in set theory. In fact Kruskal's tree theorem (or its finite form) is undecidable in a much stronger system codifying the principles acceptable based on a philosophy of mathematics called predicativism. The related but more general graph minor theorem (2003) has consequences for computational complexity theory.

6.5 Relationship with computability

See also: Halting problem § Relationship with Gödel's incompleteness theorems

The incompleteness theorem is closely related to several results about undecidable sets in recursion theory.

Stephen Cole Kleene (1943) presented a proof of Gödel's incompleteness theorem using basic results of computability theory. One such result shows that the halting problem is undecidable: there is no computer program that can correctly determine, given any program P as input, whether P eventually halts when run with a particular given input. Kleene showed that the existence of a complete effective system of arithmetic with certain consistency properties would force the halting problem to be decidable, a contradiction. This method of proof has also been presented by Shoenfield (1967, p. 132); Charlesworth (1980); and Hopcroft and Ullman (1979).

Franzén (2004, p. 73) explains how Matiyasevich's solution to Hilbert's 10th problem can be used to obtain a proof to Gödel's first incompleteness theorem. Matiyasevich proved that there is no algorithm that, given a multivariate polynomial $p(x_1, x_2,...,x_k)$ with integer coefficients, determines whether there is an integer solution to the equation $p = 0$. Because polynomials with integer coefficients, and integers themselves, are directly expressible in the language of arithmetic, if a multivariate integer polynomial equation $p = 0$ does have a solution in the integers then any sufficiently strong system of arithmetic T will prove this. Moreover, if the system T is ω-consistent, then it will never prove that a particular polynomial equation has a solution when in fact there is no solution in the integers. Thus, if T were complete and ω-consistent, it would be possible to determine algorithmically whether a polynomial equation has a solution by merely enumerating proofs of T until either "p has a solution" or "p has no solution" is found, in contradiction to Matiyasevich's theorem. Moreover, for each consistent effectively generated system T, it is possible to effectively

generate a multivariate polynomial p over the integers such that the equation $p = 0$ has no solutions over the integers, but the lack of solutions cannot be proved in T (Davis 2006:416, Jones 1980).

Smorynski (1977, p. 842) shows how the existence of recursively inseparable sets can be used to prove the first incompleteness theorem. This proof is often extended to show that systems such as Peano arithmetic are essentially undecidable (see Kleene 1967, p. 274).

Chaitin's incompleteness theorem gives a different method of producing independent sentences, based on Kolmogorov complexity. Like the proof presented by Kleene that was mentioned above, Chaitin's theorem only applies to theories with the additional property that all their axioms are true in the standard model of the natural numbers. Gödel's incompleteness theorem is distinguished by its applicability to consistent theories that nonetheless include statements that are false in the standard model; these theories are known as ω-inconsistent.

6.6 Proof sketch for the first theorem

Main article: Proof sketch for Gödel's first incompleteness theorem

The proof by contradiction has three essential parts. To begin, choose a formal system that meets the proposed criteria:

1. Statements in the system can be represented by natural numbers (known as Gödel numbers). The significance of this is that properties of statements—such as their truth and falsehood—will be equivalent to determining whether their Gödel numbers have certain properties, and that properties of the statements can therefore be demonstrated by examining their Gödel numbers. This part culminates in the construction of a formula expressing the idea that *"statement S is provable in the system"* (which can be applied to any statement "S" in the system).
2. In the formal system it is possible to construct a number whose matching statement, when interpreted, is self-referential and essentially says that it (i.e. the statement itself) is unprovable. This is done using a technique called "diagonalization" (so-called because of its origins as Cantor's diagonal argument).
3. Within the formal system this statement permits a demonstration that it is neither provable nor disprovable in the system, and therefore the system cannot in fact be ω-consistent. Hence the original assumption that the proposed system met the criteria is false.

6.6.1 Arithmetization of syntax

The main problem in fleshing out the proof described above is that it seems at first that to construct a statement p that is equivalent to "p cannot be proved", p would somehow have to contain a reference to p, which could easily give rise to an infinite regress. Gödel's ingenious technique is to show that statements can be matched with numbers (often called the arithmetization of syntax) in such a way that *"proving a statement"* can be replaced with *"testing whether a number has a given property"*. This allows a self-referential formula to be constructed in a way that avoids any infinite regress of definitions. The same technique was later used by Alan Turing in his work on the Entscheidungsproblem.

In simple terms, a method can be devised so that every formula or statement that can be formulated in the system gets a unique number, called its **Gödel number**, in such a way that it is possible to mechanically convert back and forth between formulas and Gödel numbers. The numbers involved might be very long indeed (in terms of number of digits), but this is not a barrier; all that matters is that such numbers can be constructed. A simple example is the way in which English is stored as a sequence of numbers in computers using ASCII or Unicode:

- The word **HELLO** is represented by 72-69-76-76-79 using decimal ASCII, i.e. the number 7269767679.
- The logical statement **x=y => y=x** is represented by 120-061-121-032-061-062-032-121-061-120 using octal ASCII, i.e. the number 120061121032061062032121061120.

In principle, proving a statement true or false can be shown to be equivalent to proving that the number matching the statement does or doesn't have a given property. Because the formal system is strong enough to support reasoning

about *numbers in general*, it can support reasoning about *numbers that represent formulae and statements* as well. Crucially, because the system can support reasoning about *properties of numbers*, the results are equivalent to reasoning about *provability of their equivalent statements*.

6.6.2 Construction of a statement about "provability"

Having shown that in principle the system can indirectly make statements about provability, by analyzing properties of those numbers representing statements it is now possible to show how to create a statement that actually does this.

A formula $F(x)$ that contains exactly one free variable x is called a *statement form* or *class-sign*. As soon as x is replaced by a specific number, the statement form turns into a *bona fide* statement, and it is then either provable in the system, or not. For certain formulas one can show that for every natural number n, F(n) is true if and only if it can be proved (the precise requirement in the original proof is weaker, but for the proof sketch this will suffice). In particular, this is true for every specific arithmetic operation between a finite number of natural numbers, such as "2×3=6".

Statement forms themselves are not statements and therefore cannot be proved or disproved. But every statement form $F(x)$ can be assigned a Gödel number denoted by $\mathbf{G}(F)$. The choice of the free variable used in the form $F(x)$ is not relevant to the assignment of the Gödel number $\mathbf{G}(F)$.

The notion of provability itself can also be encoded by Gödel numbers, in the following way: since a proof is a list of statements which obey certain rules, the Gödel number of a proof can be defined. Now, for every statement p, one may ask whether a number x is the Gödel number of its proof. The relation between the Gödel number of p and x, the potential Gödel number of its proof, is an arithmetical relation between two numbers. Therefore, there is a statement form Bew(y) that uses this arithmetical relation to state that a Gödel number of a proof of y exists:

> Bew(y) = $\exists\, x$ (y is the Gödel number of a formula and x is the Gödel number of a proof of the formula encoded by y).

The name **Bew** is short for *beweisbar*, the German word for "provable"; this name was originally used by Gödel to denote the provability formula just described. Note that "Bew(y)" is merely an abbreviation that represents a particular, very long, formula in the original language of T; the string "Bew" itself is not claimed to be part of this language.

An important feature of the formula Bew(y) is that if a statement p is provable in the system then Bew($\mathbf{G}(p)$) is also provable. This is because any proof of p would have a corresponding Gödel number, the existence of which causes Bew($\mathbf{G}(p)$) to be satisfied.

6.6.3 Diagonalization

The next step in the proof is to obtain a statement that says it is unprovable. Although Gödel constructed this statement directly, the existence of at least one such statement follows from the diagonal lemma, which says that for any sufficiently strong formal system and any statement form F there is a statement p such that the system proves

$$p \leftrightarrow F(\mathbf{G}(p)).$$

By letting F be the negation of Bew(x), we obtain the theorem

$$p \leftrightarrow \sim Bew(\mathbf{G}(p))$$

and the p defined by this roughly states that its own Gödel number is the Gödel number of an unprovable formula.

The statement p is not literally equal to ~Bew($\mathbf{G}(p)$); rather, p states that if a certain calculation is performed, the resulting Gödel number will be that of an unprovable statement. But when this calculation is performed, the resulting Gödel number turns out to be the Gödel number of p itself. This is similar to the following sentence in English:

> ", when preceded by itself in quotes, is unprovable.", when preceded by itself in quotes, is unprovable.

This sentence does not directly refer to itself, but when the stated transformation is made the original sentence is obtained as a result, and thus this sentence asserts its own unprovability. The proof of the diagonal lemma employs a similar method.

Now, assume that the axiomatic system is ω-consistent, and let p be the statement obtained in the previous section.

If p were provable, then Bew($\mathbf{G}(p)$) would be provable, as argued above. But p asserts the negation of Bew($\mathbf{G}(p)$). Thus the system would be inconsistent, proving both a statement and its negation. This contradiction shows that p cannot be provable.

If the negation of p were provable, then Bew($\mathbf{G}(p)$) would be provable (because p was constructed to be equivalent to the negation of Bew($\mathbf{G}(p)$)). However, for each specific number x, x cannot be the Gödel number of the proof of p, because p is not provable (from the previous paragraph). Thus on one hand the system proves there is a number with a certain property (that it is the Gödel number of the proof of p), but on the other hand, for every specific number x, we can prove that it does not have this property. This is impossible in an ω-consistent system. Thus the negation of p is not provable.

Thus the statement p is undecidable in our axiomatic system: it can neither be proved nor disproved within the system.

In fact, to show that p is not provable only requires the assumption that the system is consistent. The stronger assumption of ω-consistency is required to show that the negation of p is not provable. Thus, if p is constructed for a particular system:

- If the system is ω-consistent, it can prove neither p nor its negation, and so p is undecidable.
- If the system is consistent, it may have the same situation, or it may prove the negation of p. In the later case, we have a statement ("not p") which is false but provable, and the system is not ω-consistent.

If one tries to "add the missing axioms" to avoid the incompleteness of the system, then one has to add either p or "not p" as axioms. But then the definition of "being a Gödel number of a proof" of a statement changes. which means that the formula Bew(x) is now different. Thus when we apply the diagonal lemma to this new Bew, we obtain a new statement p, different from the previous one, which will be undecidable in the new system if it is ω-consistent.

6.6.4 Proof via Berry's paradox

George Boolos (1989) sketches an alternative proof of the first incompleteness theorem that uses Berry's paradox rather than the liar paradox to construct a true but unprovable formula. A similar proof method was independently discovered by Saul Kripke (Boolos 1998, p. 383). Boolos's proof proceeds by constructing, for any computably enumerable set S of true sentences of arithmetic, another sentence which is true but not contained in S. This gives the first incompleteness theorem as a corollary. According to Boolos, this proof is interesting because it provides a "different sort of reason" for the incompleteness of effective, consistent theories of arithmetic (Boolos 1998, p. 388).

6.6.5 Computer verified proofs

See also: Automated theorem proving

The incompleteness theorems are among a relatively small number of nontrivial theorems that have been transformed into formalized theorems that can be completely verified by proof assistant software. Gödel's original proofs of the incompleteness theorems, like most mathematical proofs, were written in natural language intended for human readers.

Computer-verified proofs of versions of the first incompleteness theorem were announced by Natarajan Shankar in 1986 using Nqthm (Shankar 1994), by Russell O'Connor in 2003 using Coq (O'Connor 2005) and by John Harrison in 2009 using HOL Light (Harrison 2009). A computer-verified proof of both incompleteness theorems was announced by Lawrence Paulson in 2013 using Isabelle (Paulson 2014).

6.7 Proof sketch for the second theorem

The main difficulty in proving the second incompleteness theorem is to show that various facts about provability used in the proof of the first incompleteness theorem can be formalized within the system using a formal predicate for provability. Once this is done, the second incompleteness theorem follows by formalizing the entire proof of the first incompleteness theorem within the system itself.

Let p stand for the undecidable sentence constructed above, and assume that the consistency of the system can be proved from within the system itself. The demonstration above shows that if the system is consistent, then p is not provable. The proof of this implication can be formalized within the system, and therefore the statement "p is not provable", or "not $P(p)$" can be proved in the system.

But this last statement is equivalent to p itself (and this equivalence can be proved in the system), so p can be proved in the system. This contradiction shows that the system must be inconsistent.

6.8 Discussion and implications

The incompleteness results affect the philosophy of mathematics, particularly versions of formalism, which use a single system of formal logic to define their principles.

6.8.1 Consequences for logicism and Hilbert's second problem

The incompleteness theorem is sometimes thought to have severe consequences for the program of logicism proposed by Gottlob Frege and Bertrand Russell, which aimed to define the natural numbers in terms of logic (Hellman 1981, p. 451–468). Bob Hale and Crispin Wright argue that it is not a problem for logicism because the incompleteness theorems apply equally to first order logic as they do to arithmetic. They argue that only those who believe that the natural numbers are to be defined in terms of first order logic have this problem.

Many logicians believe that Gödel's incompleteness theorems struck a fatal blow to David Hilbert's second problem, which asked for a finitary consistency proof for mathematics. The second incompleteness theorem, in particular, is often viewed as making the problem impossible. Not all mathematicians agree with this analysis, however, and the status of Hilbert's second problem is not yet decided (see "Modern viewpoints on the status of the problem").

6.8.2 Minds and machines

Main article: Mechanism (philosophy) § Gödelian arguments

Authors including the philosopher J. R. Lucas and physicist Roger Penrose have debated what, if anything, Gödel's incompleteness theorems imply about human intelligence. Much of the debate centers on whether the human mind is equivalent to a Turing machine, or by the Church–Turing thesis, any finite machine at all. If it is, and if the machine is consistent, then Gödel's incompleteness theorems would apply to it.

Hilary Putnam (1960) suggested that while Gödel's theorems cannot be applied to humans, since they make mistakes and are therefore inconsistent, it may be applied to the human faculty of science or mathematics in general. Assuming that it is consistent, either its consistency cannot be proved or it cannot be represented by a Turing machine.

Avi Wigderson (2010) has proposed that the concept of mathematical "knowability" should be based on computational complexity rather than logical decidability. He writes that "when *knowability* is interpreted by modern standards, namely via computational complexity, the Gödel phenomena are very much with us."

6.8.3 Paraconsistent logic

Although Gödel's theorems are usually studied in the context of classical logic, they also have a role in the study of paraconsistent logic and of inherently contradictory statements (*dialetheia*). Graham Priest (1984, 2006) argues that replacing the notion of formal proof in Gödel's theorem with the usual notion of informal proof can be used to show that naive mathematics is inconsistent, and uses this as evidence for dialetheism. The cause of this inconsistency is

the inclusion of a truth predicate for a system within the language of the system (Priest 2006:47). Stewart Shapiro (2002) gives a more mixed appraisal of the applications of Gödel's theorems to dialetheism.

6.8.4 Appeals to the incompleteness theorems in other fields

Appeals and analogies are sometimes made to the incompleteness theorems in support of arguments that go beyond mathematics and logic. Several authors have commented negatively on such extensions and interpretations, including Torkel Franzén (2004); Alan Sokal and Jean Bricmont (1999); and Ophelia Benson and Jeremy Stangroom (2006). Bricmont and Stangroom (2006, p. 10), for example, quote from Rebecca Goldstein's comments on the disparity between Gödel's avowed Platonism and the anti-realist uses to which his ideas are sometimes put. Sokal and Bricmont (1999, p. 187) criticize Régis Debray's invocation of the theorem in the context of sociology; Debray has defended this use as metaphorical (ibid.).

6.9 History

After Gödel published his proof of the completeness theorem as his doctoral thesis in 1929, he turned to a second problem for his habilitation. His original goal was to obtain a positive solution to Hilbert's second problem (Dawson 1997, p. 63). At the time, theories of the natural numbers and real numbers similar to second-order arithmetic were known as "analysis", while theories of the natural numbers alone were known as "arithmetic".

Gödel was not the only person working on the consistency problem. Ackermann had published a flawed consistency proof for analysis in 1925, in which he attempted to use the method of ε-substitution originally developed by Hilbert. Later that year, von Neumann was able to correct the proof for a system of arithmetic without any axioms of induction. By 1928, Ackermann had communicated a modified proof to Bernays; this modified proof led Hilbert to announce his belief in 1929 that the consistency of arithmetic had been demonstrated and that a consistency proof of analysis would likely soon follow. After the publication of the incompleteness theorems showed that Ackermann's modified proof must be erroneous, von Neumann produced a concrete example showing that its main technique was unsound (Zach 2006, p. 418, Zach 2003, p. 33).

In the course of his research, Gödel discovered that although a sentence which asserts its own falsehood leads to paradox, a sentence that asserts its own non-provability does not. In particular, Gödel was aware of the result now called Tarski's indefinability theorem, although he never published it. Gödel announced his first incompleteness theorem to Carnap, Feigel and Waismann on August 26, 1930; all four would attend a key conference in Königsberg the following week.

6.9.1 Announcement

The 1930 Königsberg conference was a joint meeting of three academic societies, with many of the key logicians of the time in attendance. Carnap, Heyting, and von Neumann delivered one-hour addresses on the mathematical philosophies of logicism, intuitionism, and formalism, respectively (Dawson 1996, p. 69). The conference also included Hilbert's retirement address, as he was leaving his position at the University of Göttingen. Hilbert used the speech to argue his belief that all mathematical problems can be solved. He ended his address by saying,

> For the mathematician there is no *Ignorabimus*, and, in my opinion, not at all for natural science either. ... The true reason why [no one] has succeeded in finding an unsolvable problem is, in my opinion, that there is no unsolvable problem. In contrast to the foolish *Ignoramibus*, our credo avers: We must know. We shall know!

This speech quickly became known as a summary of Hilbert's beliefs on mathematics (its final six words, "*Wir müssen wissen. Wir werden wissen!*", were used as Hilbert's epitaph in 1943). Although Gödel was likely in attendance for Hilbert's address, the two never met face to face (Dawson 1996, p. 72).

Gödel announced his first incompleteness theorem at a roundtable discussion session on the third day of the conference. The announcement drew little attention apart from that of von Neumann, who pulled Gödel aside for conversation. Later that year, working independently with knowledge of the first incompleteness theorem, von Neumann obtained a proof of the second incompleteness theorem, which he announced to Gödel in a letter dated November

20, 1930 (Dawson 1996, p. 70). Gödel had independently obtained the second incompleteness theorem and included it in his submitted manuscript, which was received by *Monatshefte für Mathematik* on November 17, 1930.

Gödel's paper was published in the *Monatshefte* in 1931 under the title *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I* (On Formally Undecidable Propositions in Principia Mathematica and Related Systems I). As the title implies, Gödel originally planned to publish a second part of the paper; it was never written.

6.9.2 Generalization and acceptance

Gödel gave a series of lectures on his theorems at Princeton in 1933–1934 to an audience that included Church, Kleene, and Rosser. By this time, Gödel had grasped that the key property his theorems required is that the system must be effective (at the time, the term "general recursive" was used). Rosser proved in 1936 that the hypothesis of ω-consistency, which was an integral part of Gödel's original proof, could be replaced by simple consistency, if the Gödel sentence was changed in an appropriate way. These developments left the incompleteness theorems in essentially their modern form.

Gentzen published his consistency proof for first-order arithmetic in 1936. Hilbert accepted this proof as "finitary" although (as Gödel's theorem had already shown) it cannot be formalized within the system of arithmetic that is being proved consistent.

The impact of the incompleteness theorems on Hilbert's program was quickly realized. Bernays included a full proof of the incompleteness theorems in the second volume of *Grundlagen der Mathematik* (1939), along with additional results of Ackermann on the ε-substitution method and Gentzen's consistency proof of arithmetic. This was the first full published proof of the second incompleteness theorem.

6.9.3 Criticisms

Finsler

Paul Finsler (1926) used a version of Richard's paradox to construct an expression that was false but unprovable in a particular, informal framework he had developed. Gödel was unaware of this paper when he proved the incompleteness theorems (Collected Works Vol. IV., p. 9). Finsler wrote to Gödel in 1931 to inform him about this paper, which Finsler felt had priority for an incompleteness theorem. Finsler's methods did not rely on formalized provability, and had only a superficial resemblance to Gödel's work (van Heijenoort 1967:328). Gödel read the paper but found it deeply flawed, and his response to Finsler laid out concerns about the lack of formalization (Dawson:89). Finsler continued to argue for his philosophy of mathematics, which eschewed formalization, for the remainder of his career.

Zermelo

In September 1931, Ernst Zermelo wrote Gödel to announce what he described as an "essential gap" in Gödel's argument (Dawson:76). In October, Gödel replied with a 10-page letter (Dawson:76, Grattan-Guinness:512-513), where he pointed out that Zermelo mistakenly assumed that the notion of truth in a system is definable in that system (which is not true in general by Tarski's undefinability theorem). But Zermelo did not relent and published his criticisms in print with "a rather scathing paragraph on his young competitor" (Grattan-Guinness:513). Gödel decided that to pursue the matter further was pointless, and Carnap agreed (Dawson:77). Much of Zermelo's subsequent work was related to logics stronger than first-order logic, with which he hoped to show both the consistency and categoricity of mathematical theories.

Wittgenstein

Ludwig Wittgenstein wrote several passages about the incompleteness theorems that were published posthumously in his 1953 *Remarks on the Foundations of Mathematics*, in particular one section sometimes called the "notorious paragraph" where he seems to confuse the notions of "true" and "provable" in Russell's system. Gödel was a member of the Vienna Circle during the period in which Wittgenstein's early ideal language philosophy and Tractatus Logico-Philosophicus dominated the circle's thinking. There has been some controversy about whether Wittgenstein misunderstood the incompletelness theorem or just expressed himself unclearly. Writings in Gödel's Nachlass

express the belief that Wittgenstein misread his ideas.

Multiple commentators have read Wittgenstein as misunderstanding Gödel (Rodych 2003), although Juliet Floyd and Hilary Putnam (2000), as well as Graham Priest (2004) have provided textual readings arguing that most commentary misunderstands Wittgenstein. On their release, Bernays, Dummett, and Kreisel wrote separate reviews on Wittgenstein's remarks, all of which were extremely negative (Berto 2009:208). The unanimity of this criticism caused Wittgenstein's remarks on the incompleteness theorems to have little impact on the logic community. In 1972, Gödel stated: "Has Wittgenstein lost his mind? Does he mean it seriously? He intentionally utters trivially nonsensical statements" (Wang 1996:179), and wrote to Karl Menger that Wittgenstein's comments demonstrate a misunderstanding of the incompleteness theorems writing:

> "It is clear from the passages you cite that Wittgenstein did *not* understand [the first incompleteness theorem] (or pretended not to understand it). He interpreted it as a kind of logical paradox, while in fact is just the opposite, namely a mathematical theorem within an absolutely uncontroversial part of mathematics (finitary number theory or combinatorics)." (Wang 1996:179)

Since the publication of Wittgenstein's *Nachlass* in 2000, a series of papers in philosophy have sought to evaluate whether the original criticism of Wittgenstein's remarks was justified. Floyd and Putnam (2000) argue that Wittgenstein had a more complete understanding of the incompleteness theorem than was previously assumed. They are particularly concerned with the interpretation of a Gödel sentence for an ω-inconsistent system as actually saying "I am not provable", since the system has no models in which the provability predicate corresponds to actual provability. Rodych (2003) argues that their interpretation of Wittgenstein is not historically justified, while Bays (2004) argues against Floyd and Putnam's philosophical analysis of the provability predicate. Berto (2009) explores the relationship between Wittgenstein's writing and theories of paraconsistent logic.

6.10 See also

- Gödel's completeness theorem
- Gödel's speed-up theorem
- *Gödel, Escher, Bach*
- Löb's Theorem
- *Minds, Machines and Gödel*
- Münchhausen trilemma
- Non-standard model of arithmetic
- Proof theory
- Provability logic
- Quining
- Tarski's undefinability theorem
- Theory of everything#Gödel's incompleteness theorem
- Third Man Argument

6.11 Notes

[1] Gödel's words are: ". . . my proof is applicable to *every* formal system containing arithmetic", appearing in Letter 3. Gödel to [Ernest] Nagel dated March 14, 1957; page 147 in Kurt Gödel, 2003 and 2014, "Collected Works Volume V: Correspondence H-V", Clarendon Press, Oxford UK, ISBN 9780191003776. Also cf introductory comments written by Charles Parsons and Wilfried Seig page 136.

[2] The Roman numeral "I" indicates that Gödel intended to publish a sequel but "The prompt acceptance of his results was one of the reasons that made him change his plan", cf the text and its footnote 68a in van Heijenoort 1967:616

6.12 References

6.12.1 Articles by Gödel

- Kurt Gödel, 1931, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", Monatshefte für Mathematik und Physik, v. 38 n. 1, pp. 173–198.

- —, 1931, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", in Solomon Feferman, ed., 1986. *Kurt Gödel Collected works, Vol. I*. Oxford University Press, pp. 144–195. ISBN 978-0195147209. The original German with a facing English translation, preceded by an introductory note by Stephen Cole Kleene.

- —, 1951, "Some basic theorems on the foundations of mathematics and their implications", in Solomon Feferman, ed., 1995. *Kurt Gödel Collected works, Vol. III*, Oxford University Press, pp. 304–323. ISBN 978-0195147223.

6.12.2 Translations, during his lifetime, of Gödel's paper into English

None of the following agree in all translated words and in typography. The typography is a serious matter, because Gödel expressly wished to emphasize "those metamathematical notions that had been defined in their usual sense before . . ." (van Heijenoort 1967:595). Three translations exist. Of the first John Dawson states that: "The Meltzer translation was seriously deficient and received a devastating review in the *Journal of Symbolic Logic*; "Gödel also complained about Braithwaite's commentary (Dawson 1997:216). "Fortunately, the Meltzer translation was soon supplanted by a better one prepared by Elliott Mendelson for Martin Davis's anthology *The Undecidable* . . . he found the translation "not quite so good" as he had expected . . . [but because of time constraints he] agreed to its publication" (ibid). (In a footnote Dawson states that "he would regret his compliance, for the published volume was marred throughout by sloppy typography and numerous misprints" (ibid)). Dawson states that "The translation that Gödel favored was that by Jean van Heijenoort" (ibid). For the serious student another version exists as a set of lecture notes recorded by Stephen Kleene and J. B. Rosser "during lectures given by Gödel at to the Institute for Advanced Study during the spring of 1934" (cf commentary by Davis 1965:39 and beginning on p. 41); this version is titled "On Undecidable Propositions of Formal Mathematical Systems". In their order of publication:

- B. Meltzer (translation) and R. B. Braithwaite (Introduction), 1962. *On Formally Undecidable Propositions of Principia Mathematica and Related Systems*, Dover Publications, New York (Dover edition 1992), ISBN 0-486-66980-7 (pbk.) This contains a useful translation of Gödel's German abbreviations on pp. 33–34. As noted above, typography, translation and commentary is suspect. Unfortunately, this translation was reprinted with all its suspect content by

    - Stephen Hawking editor, 2005. *God Created the Integers: The Mathematical Breakthroughs That Changed History*, Running Press, Philadelphia, ISBN 0-7624-1922-9. Gödel's paper appears starting on p. 1097, with Hawking's commentary starting on p. 1089.

- Martin Davis editor, 1965. *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable problems and Computable Functions*, Raven Press, New York, no ISBN. Gödel's paper begins on page 5, preceded by one page of commentary.

- Jean van Heijenoort editor, 1967, 3rd edition 1967. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*, Harvard University Press, Cambridge Mass., ISBN 0-674-32449-8 (pbk). van Heijenoort did the translation. He states that "Professor Gödel approved the translation, which in many places was accommodated to his wishes." (p. 595). Gödel's paper begins on p. 595; van Heijenoort's commentary begins on p. 592.

- Martin Davis editor, 1965, ibid. "On Undecidable Propositions of Formal Mathematical Systems." A copy with Gödel's corrections of errata and Gödel's added notes begins on page 41, preceded by two pages of Davis's commentary. Until Davis included this in his volume this lecture existed only as mimeographed notes.

6.12.3 Articles by others

- George Boolos, 1989, "A New Proof of the Gödel Incompleteness Theorem", *Notices of the American Mathematical Society*, v, 36, pp. 388–390 and p. 676, reprinted in Boolos, 1998, *Logic, Logic, and Logic*, Harvard Univ. Press. ISBN 0-674-53766-1
- Bernd Buldt, 2014, "The Scope of Gödel's First Incompleteness Theorem", *Logica Universalis*, v. 8, pp. 499–552. doi:10.1007/s11787-014-0107-3
- Arthur Charlesworth, 1980, "A Proof of Godel's Theorem in Terms of Computer Programs", *Mathematics Magazine*, v. 54 n. 3, pp. 109–121. JStor
- Martin Davis, 2006, "The Incompleteness Theorem", *Notices of the AMS*, v. 53 n. 4, pp. 414.
- Jean van Heijenoort, 1963, "Gödel's Theorem" in Edwards, Paul, ed., *Encyclopedia of Philosophy, v. 3*. Macmillan, pp. 348–57.
- Geoffrey Hellman, 1981, "How to Gödel a Frege-Russell: Gödel's Incompleteness Theorems and Logicism", *Noûs*, v. 15 n. 4, Special Issue on Philosophy of Mathematics, pp. 451–468.
- David Hilbert, 1900, "Mathematical Problems." English translation of a lecture delivered before the International Congress of Mathematicians at Paris, containing Hilbert's statement of his Second Problem.
- Martin Hirzel, 2000, *On formally undecidable propositions of Principia Mathematica and related systems I.*. An English translation of Gödel's paper.
- Makoto Kikuchi and Kazayuki Tanaka, 1994, "On formalization of model-theoretic proofs of Gödel's theorems",*Notre Dame Journal of Formal Logic*, v. 5 n. 3, pp. 403–412. doi:10.1305/ndjfl/1040511346 MR 1326122
- Stephen Cole Kleene, 1943, "Recursive predicates and quantifiers", reprinted from *Transactions of the American Mathematical Society*, v. 53 n. 1, pp. 41–73 in Martin Davis 1965, *The Undecidable* (loc. cit.) pp. 255–287.
- Panu Raatikainen, 2015, "Gödel's Incompleteness Theorems", Stanford Encyclopedia of Philosophy. Accessed April 3, 2015.
- John Barkley Rosser, 1936, "Extensions of some theorems of Gödel and Church", reprinted from the *Journal of Symbolic Logic*, v. 1 (1936) pp. 87–91, in Martin Davis 1965, *The Undecidable* (loc. cit.) pp. 230–235.
- —, 1939, "An Informal Exposition of proofs of Gödel's Theorem and Church's Theorem", Reprinted from the *Journal of Symbolic Logic*, v. 4 (1939) pp. 53–60, in Martin Davis 1965, *The Undecidable* (loc. cit.) pp. 223–230
- C. Smoryński, 1982, "The incompleteness theorems", in Jon Barwise, ed., *Handbook of Mathematical Logic*, North-Holland, pp. 821–866. ISBN 978-0-444-86388-1
- Dan E. Willard, 2001, "Self-Verifying Axiom Systems, the Incompleteness Theorem and Related Reflection Principles", *Journal of Symbolic Logic*, v. 66 n. 2, pp. 536–596. doi:10.2307/2695030
- Richard Zach, 2003. "The Practice of Finitism: Epsilon Calculus and Consistency Proofs in Hilbert's Program" *Synthese*, v. 137 n. 1, pp. 211–259. doi:10.1023/A:1026247421383
- —, 2005, "Kurt Gödel, Paper on the incompleteness theorems" in Ivor Grattan-Guinness, ed. *Landmark Writings in Western Mathematics*, Elsevier, pp. 917–925. doi:10.1016/B978-044450871-3/50152-2

6.12.4 Books about the theorems

- Francesco Berto. *There's Something about Gödel: The Complete Guide to the Incompleteness Theorem* John Wiley and Sons. 2010.
- Norbert Domeisen, 1990. *Logik der Antinomien*. Bern: Peter Lang. 142 S. 1990. ISBN 3-261-04214-1. Zentralblatt MATH

- Torkel Franzén, 2004. *Gödel's Theorem: An Incomplete Guide to its Use and Abuse*. A.K. Peters. ISBN 1-56881-238-8 MR 2007d:03001
- Douglas Hofstadter, 1979. *Gödel, Escher, Bach: An Eternal Golden Braid*. Vintage Books. ISBN 0-465-02685-0. 1999 reprint: ISBN 0-465-02656-7. MR 80j:03009
- —, 2007. *I Am a Strange Loop*. Basic Books. ISBN 978-0-465-03078-1. ISBN 0-465-03078-5. MR 2008g:00004
- Stanley Jaki, OSB, 2005. *The drama of the quantities*. Real View Books.
- Per Lindström, 1997. *Aspects of Incompleteness*, Lecture Notes in Logic v. 10.
- J.R. Lucas, FBA, 1970. *The Freedom of the Will*. Clarendon Press, Oxford, 1970.
- Ernest Nagel, James Roy Newman, Douglas Hofstadter, 2002 (1958). *Gödel's Proof*, revised ed. ISBN 0-8147-5816-9. MR 2002i:03001
- Rudy Rucker, 1995 (1982). *Infinity and the Mind: The Science and Philosophy of the Infinite*. Princeton Univ. Press. MR 84d:03012
- Peter Smith, 2007. *An Introduction to Gödel's Theorems.* Cambridge University Press. MathSciNet
- N. Shankar, 1994. *Metamathematics, Machines and Gödel's Proof*, Volume 38 of Cambridge tracts in theoretical computer science. ISBN 0-521-58533-3
- Raymond Smullyan, 1987. *Forever Undecided* ISBN 0192801414 - puzzles based on undecidability in formal systems
- —, 1991. *Godel's Incompleteness Theorems*. Oxford Univ. Press.
- —, 1994. *Diagonalization and Self-Reference*. Oxford Univ. Press. MR 96c:03001
- —, 2013. *The Godelian Puzzle Book: Puzzles, Paradoxes and Proofs*. Courier Corporation. ISBN 978-0-486-49705-1.
- Hao Wang, 1997. *A Logical Journey: From Gödel to Philosophy*. MIT Press. ISBN 0-262-23189-1 MR 97m:01090

6.12.5 Miscellaneous references

- Francesco Berto, 2009, "The Gödel Paradox and Wittgenstein's Reasons" *Philosophia Mathematica* (III) 17.
- John W. Dawson, Jr., 1997. *Logical Dilemmas: The Life and Work of Kurt Gödel*, A. K. Peters, Wellesley Mass, ISBN 1-56881-256-6.
- Rebecca Goldstein, 2005, *Incompleteness: the Proof and Paradox of Kurt Gödel*, W. W. Norton & Company. ISBN 0-393-05169-2
- Juliet Floyd and Hilary Putnam, 2000, "A Note on Wittgenstein's 'Notorious Paragraph' About the Gödel Theorem", *Journal of Philosophy* v. 97 n. 11, pp. 624–632.
- John Harrison, 2009, "Handbook of Practical Logic and Automated Reasoning", Cambridge University Press, ISBN 0521899575
- David Hilbert and Paul Bernays, *Grundlagen der Mathematik*, Springer-Verlag.
- John Hopcroft and Jeffrey Ullman 1979, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, ISBN 0-201-02988-X
- James P. Jones, *Undecidable Diophantine Equations*, *Bulletin of the American Mathematical Society*, v. 3 n. 2, 1980, pp. 859–862.
- Stephen Cole Kleene, 1967, *Mathematical Logic*. Reprinted by Dover, 2002. ISBN 0-486-42533-9

- Russell O'Connor, 2005, "Essential Incompleteness of Arithmetic Verified by Coq", Lecture Notes in Computer Science v. 3603, pp. 245–260.
- Lawrence Paulson, 2013, "A machine-assisted proof of Gödel's incompleteness theorems for the theory of hereditarily finite sets", *Review of Symbolic Logic*, v. 7 n. 3, 484–498.
- Graham Priest, 1984, "Logic of Paradox Revisited", *Journal of Philosophical Logic*, v. 13,` n. 2, pp. 153–179.
- —, 2004, *Wittgenstein's Remarks on Gödel's Theorem* in Max Kölbel, ed., *Wittgenstein's lasting significance*, Psychology Press, pp. 207–227.
- —, 2006, *In Contradiction: A Study of the Transconsistent*, Oxford University Press, ISBN 0-19-926329-9
- Hilary Putnam, 1960, *Minds and Machines* in Sidney Hook, ed., *Dimensions of Mind: A Symposium*. New York University Press. Reprinted in Anderson, A. R., ed., 1964. *Minds and Machines*. Prentice-Hall: 77.
- Wolfgang Rautenberg, 2010, *A Concise Introduction to Mathematical Logic*, 3rd. ed., Springer, ISBN 978-1-4419-1220-6
- Victor Rodych, 2003, "Misunderstanding Gödel: New Arguments about Wittgenstein and New Remarks by Wittgenstein", *Dialectica* v. 57 n. 3, pp. 279–313. doi:10.1111/j.1746-8361.2003.tb00272.x
- Stewart Shapiro, 2002, "Incompleteness and Inconsistency", *Mind*, v. 111, pp 817–32. doi:10.1093/mind/111.444.817
- Alan Sokal and Jean Bricmont, 1999, *Fashionable Nonsense: Postmodern Intellectuals' Abuse of Science*, Picador. ISBN 0-312-20407-8
- Joseph R. Shoenfield (1967), *Mathematical Logic*. Reprinted by A.K. Peters for the Association for Symbolic Logic, 2001. ISBN 978-1-56881-135-2
- Jeremy Stangroom and Ophelia Benson, *Why Truth Matters*, Continuum. ISBN 0-8264-9528-1
- George Tourlakis, *Lectures in Logic and Set Theory, Volume 1, Mathematical Logic*, Cambridge University Press, 2003. ISBN 978-0-521-75373-9
- Avi Wigderson, 2010, "The Gödel Phenomena in Mathematics: A Modern View", in *Kurt Gödel and the Foundations of Mathematics: Horizons of Truth*, Cambridge University Press.
- Hao Wang, 1996, *A Logical Journey: From Gödel to Philosophy*, The MIT Press, Cambridge MA, ISBN 0-262-23189-1.
- Richard Zach, 2006, "Hilbert's program then and now", in *Philosophy of Logic*, Dale Jacquette (ed.), Handbook of the Philosophy of Science, v. 5., Elsevier, pp. 411–447.

6.13 External links

-
- Godel's Incompleteness Theorems on *In Our Time* at the BBC. (listen now)
- Kennedy, Juliette (2015). "Kurt Gödel". *Stanford Encyclopedia of Philosophy*.
- Raatikainen, Panu (2015). "Gödel's Incompleteness Theorems". *Stanford Encyclopedia of Philosophy*.
- *Paraconsistent Logic § Arithmetic and Gödel's Theorem* at the Stanford Encyclopedia of Philosophy.
- MacTutor biographies:
  - Kurt Gödel.
  - Gerhard Gentzen.
  - What is Mathematics:Gödel'{ }s Theorem and Around by *Karlis Podnieks*. An online free book.
- World's shortest explanation of Gödel's theorem using a printing machine as an example.

- October 2011 RadioLab episode about/including Gödel's Incompleteness theorem
- Hazewinkel, Michiel, ed. (2001), "Gödel incompleteness theorem", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4

Chapter 7

Proof sketch for Gödel's first incompleteness theorem

This article gives a sketch of a proof of **Gödel's first incompleteness theorem**. This theorem applies to any formal theory that satisfies certain technical hypotheses, which are discussed as needed during the sketch. We will assume for the remainder of the article that a fixed theory satisfying these hypotheses has been selected.

Throughout this article the word "number" refers to a natural number. The key property these numbers possess is that any natural number can be obtained by starting with the number 0 and adding 1 a finite number of times.

7.1 Hypotheses of the theory

Gödel's theorem applies to any formal theory that satisfies certain properties. Each formal theory has a signature that specifies the nonlogical symbols in the language of the theory. For simplicity, we will assume that the language of the theory consists of:

- A constant symbol 0.
- A unary function symbol S for the successor operation and binary function symbols + and × for addition and multiplication.
- Symbols for logical conjunction, $\wedge$, disjunction, $\vee$, and negation, $\neg$.
- Universal, $\forall$, and existential, $\exists$, quantifiers.
- Binary relations, = and <, for equality and order (less than).
- Left and right parentheses for establishing precedence of quantifiers.
- A single variable symbol x and a symbol * that can be used to construct additional variables of the form x*, x**, …

This is the language of Peano arithmetic. A well-formed formula is a sequence of these symbols that is formed so as to have a well-defined reading as a mathematical formula. Thus $x = SS0$ is well formed while $x = \forall +$ is not well formed. A theory is a set of well-formed formulas with no free variables.

A theory is consistent if there is no formula F such that both F and its negation are provable. ω-consistency is a stronger property than consistency. Suppose that $F(x)$ is a formula with one free variable x. In order to be ω-consistent, the theory cannot prove both $\exists m\ F(m)$ while also proving $\neg F(n)$ for each natural number n.

The theory is assumed to be effective, which means that the set of axioms must be recursively enumerable. This means that it is theoretically possible to write a computer program that, if allowed to run forever, would output the axioms of the theory one at a time and not output anything else. This requirement is necessary; there are theories that are complete, consistent, and include elementary arithmetic, but no such theory can be effective.

7.2 Outline of the proof

For a simplified outline of the proof, see Gödel's incompleteness theorems

The sketch here is broken into three parts. In the first part, each formula of the theory is assigned a number, known as a Gödel number, in a manner that allows the formula to be effectively recovered from the number. This numbering is extended to cover finite sequences of formulas. In the second part, a specific formula $PF(x, y)$ is constructed such that for any two numbers n and m, $PF(n,m)$ holds if and only if n represents a sequence of formulas that constitutes a proof of the formula that m represents. In the third part of the proof, we construct a self-referential formula that, informally, says "I am not provable", and prove that this sentence is neither provable nor disprovable within the theory.

Importantly, all the formulas in the proof can be defined by primitive recursive functions, which themselves can be defined in first-order Peano arithmetic.

7.3 Gödel numbering

The first step of the proof is to represent (well-formed) formulas of the theory, and finite lists of these formulas, as natural numbers. These numbers are called the **Gödel numbers** of the formulas.

Begin by assigning a natural number to each symbol of the language of arithmetic, similar to the manner in which the ASCII code assigns a unique binary number to each letter and certain other characters. This article will employ the following assignment, very similar to the one Douglas Hofstadter used in his *Gödel, Escher, Bach*:

The Gödel number of a formula is obtained by concatenating the Gödel numbers of each symbol making up the formula. The Gödel numbers for each symbol are separated by a zero because by design, no Gödel number of a symbol includes a 0. Hence any formula may be correctly recovered from its Gödel number. Let $G(F)$ denote the Gödel number of the formula F.

Given the above Gödel numbering, the sentence asserting that addition commutes, $\forall x\, \forall x^*\, (x + x^* = x^* + x)$ translates as the number:

626 0 262 0 626 0 262 0 163 0 362 0 262 0 112 0 262 0 163 0 111 0 262 0 163 0 112 0 262 0 323

(Spaces have been inserted on each side of every 0 only for readability; Gödel numbers are strict concatenations of decimal digits.) Not all natural numbers represent a formula. For example, the number

111 0 626 0 112 0 262

translates to "$= \forall + x$", which is not well-formed.

Because each natural number can be obtained by applying the successor operation S to 0 a finite number of times, every natural number has its own Gödel number. For example, the Gödel number corresponding to 4, $SSSS0$, is:

123 0 123 0 123 0 123 0 666.

The assignment of Gödel numbers can be extended to finite lists of formulas. To obtain the Gödel number of a list of formulas, write the Gödel numbers of the formulas in order, separating them by two consecutive zeros. Since the Gödel number of formula never contains two consecutive zeros, each formula in a list of formulas can be effectively recovered from the Gödel number for the list.

It is crucial that the formal arithmetic be capable of proving a minimum set of facts. In particular, it must be able to prove that every number has a Gödel number. A second fact that the theory must prove is that given any Gödel number of a formula $F(x)$ with one free variable x and any number m, there is a Gödel number of the formula $F(m)$ obtained by replacing all occurrences of $G(x)$ in $G(F(x))$ with $G(m)$, and that this second Gödel number can be effectively obtained from the Gödel number of F as a function of m. To see that this is in fact possible, note that given the Gödel number for F, one can recreate the original formula, make the substitution, and then find the Gödel number of the resulting formula. This is a uniform procedure.

7.4 The provability relation

Deduction rules can then be represented by binary relations on Gödel numbers of lists of formulas. In other words, suppose that there is a deduction rule D_1, by which one can move from the formulas S_1,S_2 to a new formula S. Then the relation R_1 corresponding to this deduction rule says that n is related to m (in other words, $n\ R_1 m$ holds) if n is the Gödel number of a list of formulas containing S_1 and S_2 and m is the Gödel number of the list of formulas consisting of those in the list coded by n together with S. Because each deduction rule is concrete, it is possible to effectively determine for any natural numbers n and m whether they are related by the relation.

The second stage in the proof is to use the Gödel numbering, described above, to show that the notion of provability can be expressed within the formal language of the theory. Suppose the theory has deduction rules: D_1, D_2, D_3, … . Let R_1, R_2, R_3, … be their corresponding relations, as described above.

Every provable statement is either an axiom itself, or it can be deduced from the axioms by a finite number of applications of the deduction rules. We wish to define a set of numbers P that represents all these provable statements. We define P as the minimal set consisting of all numbers in AX (representing axioms) and closed under all the relations R_1, R_2, … . This means that whenever n is in the set P and $n\ R_\mathrm{i}\ m$ for some numbers m and i, the number m is also in the set P. It is not hard to see that P represents the set of provable statements. That is, the members of P are the Gödel numbers of the provable statements.

A proof of a formula S is itself a string of mathematical statements related by particular relations (each is either an axiom or related to former statements by deduction rules), where the last statement is S. Thus one can define the Gödel number of a proof. Moreover, one may define a statement form $PF(x,y)$, which for every two numbers x and y is provable if and only if x is the Gödel number of a proof of the statement S and $y = G(S)$.

$PF(x,y)$ is in fact an arithmetical relation, just as "$x + y = 6$" is, though a (much) more complicated one. Given such a relation $R(x,y)$, for any two specific numbers n and m, either the formula $R(m,n)$, or its negation $\neg R(m,n)$, but not both, is provable. This is because the relation between these two numbers can be simply "checked". Formally this can be proven by induction, where all these possible relations (which are of infinite number) are constructed one by one. The detailed construction of the formula PF makes essential use of the assumption that the theory is effective; it would not be possible to construct this formula without such an assumption.

7.5 Self-referential formula

For every number n and every formula $F(y)$, where y is a free variable, we define $q(n, G(F))$, a relation between two numbers n and $G(F)$, such that it corresponds to the statement "n is not the Gödel number of a proof of $F(G(F))$". Here, $F(G(F))$ can be understood as F with its own Gödel number as its argument.

Note that q takes as an argument $G(F)$, the Gödel number of F. In order to prove either $q(n, G(F))$, or $\neg q(n, G(F))$, it is necessary to perform number-theoretic operations on $G(F)$ that mirror the following steps: decode the number $G(F)$ into the formula F, replace all occurrences of y in F with the number $G(F)$, and then compute the Gödel number of the resulting formula $F(G(F))$.

Note that for every specific number n and formula $F(y)$, $q(n, G(F))$ is a straightforward (though complicated) arithmetical relation between two numbers n and $G(F)$, building on the relation PF defined earlier. Further, $q(n, G(F))$ is provable if the finite list of formulas encoded by n is not a proof of $F(G(F))$, and $\neg q(n, G(F))$ is provable if the finite list of formulas encoded by n is a proof of $F(G(F))$. Given any numbers n and $G(F)$, either $q(n, G(F))$ or $\neg q(n,G(F))$ (but not both) is provable.

Any proof of $F(G(F))$ can be encoded by a Gödel number n, such that $q(n, G(F))$ does not hold. If $q(n, G(F))$ holds for all natural numbers n, then there is no proof of $F(G(F))$. In other words, $\forall y\ q(y, G(F))$, a formula about natural numbers, corresponds to "there is no proof of $F(G(F))$".

We now define the formula $P(x) = \forall y\ q(y, x)$, where x is a free variable. The formula P itself has a Gödel number $G(P)$ as does every formula.

This formula has a free variable x. Suppose we replace it with $G(F)$, the Gödel number of a formula $F(z)$, where z is a free variable. Then, $P(G(F)) = \forall y\ q(y, G(F))$ corresponds to "there is no proof of $F(G(F))$", as we have seen.

Consider the formula $P(G(P)) = \forall y, q(y, G(P))$. This formula concerning the number $G(P)$ corresponds to "there is no proof of $P(G(P))$". We have here the self-referential feature that is crucial to the proof: A formula of the formal theory that somehow relates to its own provability within that formal theory. Very informally, $P(G(P))$ says: "I am

not provable".

We will now show that neither the formula $P(G(P))$, nor its negation $\neg P(G(P))$, is provable.

Suppose $P(G(P)) = \forall y,\ q(y,\ G(P))$ is provable. Let n be the Gödel number of a proof of $P(G(P))$. Then, as seen earlier, the formula $\neg q(n,\ G(P))$ is provable. Proving both $\neg q(n,\ G(P))$ and $\forall y\ q(y,\ G(P))$ violates the consistency of the formal theory. We therefore conclude that $P(G(P))$ is not provable.

Consider any number n. Suppose $\neg q(n,\ G(P))$ is provable. Then, n must be the Gödel number of a proof of $P(G(P))$. But we have just proved that $P(G(P))$ is not provable. Since either $q(n,\ G(P))$ or $\neg q(n,\ G(P))$ must be provable, we conclude that, for all natural numbers n, $q(n,\ G(P))$ is provable.

Suppose the negation of $P(G(P))$, $\neg P(G(P)) = \exists x\ \neg\ q(x,\ G(P))$, is provable. Proving both $\exists x\ \neg q(x,\ G(P))$, and $q(n,\ G(P))$, for all natural numbers n, violates ω-consistency of the formal theory. Thus if the theory is ω-consistent, $\neg P(G(P))$ is not provable.

We have sketched a proof showing that:

For any formal, recursively enumerable (i.e. effectively generated) theory of Peano Arithmetic,

> if it is consistent, then there exists an unprovable formula (in the language of that theory).
>
> if it is ω-consistent, then there exists a formula such that both it and its negation are unprovable.

7.5.1 The truth of the Gödel sentence

The proof of Gödel's incompleteness theorem just sketched is **proof-theoretic** (also called **syntactic**) in that it shows that if certain proofs exist (a proof of $P(G(P))$ or its negation) then they can be manipulated to produce a proof of a contradiction. This makes no appeal to whether $P(G(P))$ is "true", only to whether it is provable. Truth is a **model-theoretic**, or **semantic**, concept, and is not equivalent to provability except in special cases.

By analyzing the situation of the above proof in more detail, it is possible to obtain a conclusion about the truth of $P(G(P))$ in the standard model $\mathbb{N}$ of natural numbers. As just seen, $q(n,\ G(P))$ is provable for each natural number n, and is thus true in the model $\mathbb{N}$. Therefore, within this model,

$$P(G(P)) = \forall y\, q(y, G(P))$$

holds. This is what the statement "$P(G(P))$ is true" usually refers to—the sentence is true in the intended model. It is not true in every model, however: If it were, then by Gödel's completeness theorem it would be provable, which we have just seen is not the case.

7.6 Boolos's short proof

George Boolos (1998) vastly simplified the proof of the First Theorem, if one agrees that the theorem is equivalent to:

> "There is no algorithm M whose output contains all true sentences of arithmetic and no false ones."

"Arithmetic" refers to Peano or Robinson arithmetic, but the proof invokes no specifics of either, tacitly assuming that these systems allow '<' and '×' to have their usual meanings. Boolos proves the theorem in about two pages. His proof employs the language of first-order logic, but invokes no facts about the connectives or quantifiers. The domain of discourse is the natural numbers. The Gödel sentence builds on Berry's paradox.

Let $[n]$ abbreviate n successive applications of the successor function, starting from 0. Boolos then asserts (the details are only sketched) that there exists a defined predicate Cxz that comes out true iff an arithmetic formula containing z symbols names the number x. This proof sketch contains the only mention of Gödel numbering; Boolos merely assumes that every formula can be so numbered. Here, a formula F *names* the number n iff the following is provable:

$$\forall x(F(x) \leftrightarrow x = n)$$

Boolos then defines the related predicates:

- $Bxy \leftrightarrow \exists z(z < y \wedge Cxz)$. (English: Bxy comes out true if x can be defined in fewer than y symbols):
- $Axy \leftrightarrow \neg Bxy \wedge \forall a(a < x \rightarrow Bay)$. (English: Axy comes out true if x is the smallest number not definable in fewer than y symbols. More awkwardly, Axy holds if x cannot be defined in fewer than y symbols, and all numbers less than x can be defined using fewer than y symbols);
- $Fx \leftrightarrow \exists y((y = [10] \times [k]) \wedge Axy)$. k = the number of symbols appearing in Axy.

Fx formalizes Berry's paradox. The balance of the proof, requiring but 12 lines of text, shows that the sentence $\forall x(Fx \leftrightarrow (x = [n]))$ is true for some number n, but no algorithm M will identify it as true. Hence in arithmetic, truth outruns proof. QED.

The above predicates contain the only existential quantifiers appearing in the entire proof. The '$<$' and '$\times$' appearing in these predicates are the only defined arithmetical notions the proof requires. The proof nowhere mentions recursive functions or any facts from number theory, and Boolos claims that his proof dispenses with diagonalization. For more on this proof, see Berry's paradox.

7.7 References

- 1931, *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I. Monatshefte für Mathematik und Physik 38*: 173–98.
- English translations of the preceding:
  - Jean van Heijenoort, 1967. *From Frege to Gödel: A Source Book on Mathematical Logic*. Harvard University Press: 596–616.
  - Hirzel, Martin, 2000, *On formally undecidable propositions of Principia Mathematica and related systems I.*.
- 1951, *Some basic theorems on the foundations of mathematics and their implications* in Solomon Feferman, ed., 1995. *Collected works / Kurt Gödel, Vol. III*. Oxford University Press: 304–23.
- George Boolos, 1998, "A New Proof of the Gödel Incompleteness Theorem" in Boolos, G., *Logic, Logic, and Logic*. Harvard Univ. Press.

7.8 External links

- A concise proof of Gödel's Incompleteness Theorem.

7.9 Text and image sources, contributors, and licenses

7.9.1 Text

- **First-order logic** *Source:* https://en.wikipedia.org/wiki/First-order_logic?oldid=729639028 *Contributors:* AxelBoldt, The Anome, Ben-Baker, Dwheeler, Youandme, Stevertigo, Frecklefoot, Edward, Patrick, Michael Hardy, Kwertii, Kku, Ixfd64, Chinju, Zeno Gantner, Minesweeper, Looxix~enwiki, TallJosh, Julesd, AugPi, Dpol, Jod, Nzrs0006, Charles Matthews, Timwi, Dcoetzee, Dysprosia, Greenrd, Markhurd, Hyacinth, David.Monniaux, Robbot, Fredrik, Vanden, Wikibot, Jleedev, Tea2min, Filemon, Snobot, Giftlite, Xplat, Kim Bruning, Lethe, Jorend, Guanaco, Siroxo, Gubbubu, Mmm~enwiki, Utcursch, Kusunose, Almit39, Karl-Henner, Creidieki, Urhixidur, Lucidish, Mormegil, Rich Farmbrough, Guanabot, Paul August, Bender235, Elwikipedista~enwiki, Pmetzger, Spayrard, Chalst, Nile, Rsmelt, EmilJ, Marner, Randall Holmes, Per Olofsson, Nortexoid, Spug, ToastieIL, AshtonBenson, Obradovic Goran, Mpeisenbr, Officiallyover, Msh210, Axl, Harburg, Dhruvee, Caesura, BRW, Iannigb, Omphaloscope, Apolkhanov, Bookandcoffee, Oleg Alexandrov, Kendrick Hang, Hq3473, Joriki, Velho, Kelly Martin, Linas, Ahouseholder, Ruud Koot, BD2412, SixWingedSeraph, Grammarbot, Rjwilmsi, Tizio, .digamma, MarSch, Mike Segal, Ekspiulo, R.e.b., Penumbra2000, Mathbot, Banazir, NavarroJ, Chobot, Bgwhite, Jayme, Roboto de Ajvol, Wavelength, Borgx, Michael Slone, Marcus Cyron, Meloman, Trovatore, Expensivehat, Hakeem.gadi, JE-Compton, Saric, Arthur Rubin, Netrapt, Nahaj, Katieh5584, RG2, Otto ter Haar, Jsnx, SmackBot, InverseHypercube, Brick Thrower, Yamaguchi, Slaniel, NickGarvey, Mhss, Foxjwill, Onceler, Jon Awbrey, Turms, Henning Makholm, Tesseran, Byelf2007, Lambiam, Cdills, Dbtfz, Richard L. Peterson, Cronholm144, Physis, Loadmaster, Mets501, Pezant, Phuzion, Mike Fikes, JulianMendez, Dan Gluck, Iridescent, Hilverd, Zero sharp, JRSpriggs, 8754865, CRGreathouse, CBM, Mindfruit, Gregbard, Fl, Danman3459, Blaisorblade, Julian Mendez, Juansempere, Eubulide, Malleus Fatuorum, Mojo Hand, RobHar, Nick Number, Rriegs, Klausness, Eleuther, Jirislaby, VictorAnyakin, Childoftv, Tigranes Damaskinos, JAnDbot, Ahmed saeed, Thenub314, RubyQ, Igodard, Martinkunev, Alastair Haines, Jay Gatsby, LookingGlass, A3nm, David Eppstein, Pkrecker, TechnoFaye, Avakar, Exostor, Pomte, Maurice Carbonaro, WarthogDemon, Inquam, SpigotMap, Policron, Heyitspeter, Mistercupcake, Camrn86, English Subtitle, Crowne, Voorlandt, The Tetrast, Philogo, LBehounek, Jesin, VanishedUserABC, Paradoctor, Kgoarany, RJaguar3, ConcernedScientist, Lord British, Ljf255, SouthLake, Kumioko (renamed), DesolateReality, Anchor Link Bot, Wireless99, Randomblue, CBM2, NoBu11, Francvs, Classicalecon, Phyte, NicDumZ, Jan1nad, Gherson2, Mild Bill Hiccup, Dkf11, Nanobear~enwiki, Nanmus, Watchduck, Cacadril, Hans Adler, Djk3, Willhig, Palnot, WikHead, Subversive.sound, Sameer0s, Addbot, Norman Ramsey, Histre, Pdibner, Tassedethe, דניאל ב., Snaily, Legobot, Yobot, Ht686rg90, Cloudyed, Pcap, AnakngAraw, AnomieBOT, Citation bot, TitusCarus, Grim23, Ejars, Raulshc, FrescoBot, Hobsonlane, Mark Renier, Liiiii, Citation bot 1, Tkuvho, DrilBot, I dream of horses, Sh Najd, 34jjkky, Rlcolasanti, Diannaa, Reach Out to the Truth, Lauri.pirttiaho, WildBot, Gf uip, Klbrain, Carbo1200, Be hajian, Chharvey, Sampletalk, Bulwersator, Jaseemabid, Tijfo098, Templatetypedef, ClueBot NG, Johannes Schützel, MerlIwBot, Daviddwd, BG19bot, Pacerier, Lifeformnoho, Dhruvbaldawa, Virago250, Solomon7968, Rjs.swarnkar, Sanpra1989, Deltahedron, Gabefair, Jochen Burghardt, Hoppeduppeanut, Cptwunderlich, Seppi333, Holyseven007, Wilbertcr, Threerealtrees, Immanuel Thoughtmaker, Jwinder47, Mario Castelán Castro, Purgy Purgatorio, Comp-heur-intel, Broswald, Ndesh26, Scirocco0316 and Anonymous: 255
- **Peano axioms** *Source:* https://en.wikipedia.org/wiki/Peano_axioms?oldid=728377238 *Contributors:* AxelBoldt, LC~enwiki, Zundark, The Anome, MadSurgeon, Olivier, Patrick, Michael Hardy, Dominus, Gabbe, Meekohi, Bcrowell, AugPi, Tim Retout, Schneelocke, Ideyal, Hashar, Revolver, Rzach, Charles Matthews, Timwi, Jitse Niesen, Cjmnyc, Markhurd, Sabbut, Jonhays0, Aleph4, Altenmann, Sverdrup, Sho Uemura, Snobot, Giftlite, Lethe, Dratman, Jorend, Jason Quinn, Gadfium, Sam Hocevar, Creidieki, Andreas Kaufmann, Kaustuv, PhotoBox, Discospinster, Guanabot, Murtasa, Paul August, MarkS, Kbh3rd, Chalst, EmilJ, Randall Holmes, Smalljim, Andrewbadr, JohnyDog, Alansohn, Keenan Pepper, Sligocki, Isaac, Drbreznjev, Pediddle, Linas, Frungi, Qwertyus, FreplySpang, Rjwilmsi, MarSch, Salix alba, R.e.b., Durin, Klortho, VKokielov, Jameshfisher, Chobot, DVdm, Algebraist, YurikBot, Hede2000, KSmrq, CarlHewitt, Nahallac Silverwinds, Trovatore, Jpbowen, JulesH, Arthur Rubin, Pb30, RenamedUser jaskldjslak904, SmackBot, Melchoir, Nikhilgk, Chris the speller, Jeekc, Jdthood, Decemberster~enwiki, Vanished User 0001, Cybercobra, Acepectif, Xyzzy n, Bigmantonyd, Wybot, Alexandr.Kara, Lambiam, Wvbailey, Titus III, Evildictaitor, Loadmaster, Makyen, Mets501, Rob-nick, Zero sharp, JRSpriggs, Vaughan Pratt, CRGreathouse, CmdrObot, Diegueins, CBM, Myasuda, Gregbard, Sopoforic, DumbBOT, Headbomb, Catsmoke, Nick Number, Dajagr, Yupik, Liquid-aim-bot, VectorPosse, Admc~enwiki, Hannes Eder, Erxnmedia, JAnDbot, CosineKitty, Olaf, Thenub314, Magioladitis, JoergenB, Eliko, Leyo, George963 au, CSinColo, Policron, VolkovBot, FrankEM, Nxavar, Anonymous Dissident, Friddle~enwiki, The Tetrast, Omcnew, Lambyte, GirasoleDE, SieBot, Quasirandom, Michel421, DesolateReality, Anchor Link Bot, CBM2, The Thing That Should Not Be, Hans Adler, Rulerofutumno, Marc van Leeuwen, D.M. from Ukraine, Addbot, Lightbot, דוד שי, Yobot, Dinnertimeok, AnomieBOT, Wtachi, Materialscientist, Citation bot, ArthurBot, KHirsch, Tomdo08, Omnipaedista, RibotBOT, FrescoBot, Tkuvho, Kusluj, Kiefer.Wolfowitz, נצה מ.ה., Jauhienij, Gamewizard71, Milolance, 777sms, Aperisic, KurtSchwitters, EmausBot, Slawekb, L.o.Bryan, Quondum, Scientific29, F. Weckenbarth, ClueBot NG, Wcherowi, Aero-Plex, Lanthanum-138, Frietjes, Widr, Helpful Pixie Bot, BG19bot, Ken.sailor, Lawandeconomics1, Belnapj, Russell208, BattyBot, Robur4, Jochen Burghardt, Mark viking, Rkaup, Jose Brox, KingSupernova, Ptrsinclair, Dddddb, BemusedObserver, Voodoo Bus, Tiledphyla, Slilywimp, Squarestree, Luis150902, K.k.smirnov, Baking Soda, Goodman.jam, Dualspace and Anonymous: 144
- **Zermelo–Fraenkel set theory** *Source:* https://en.wikipedia.org/wiki/Zermelo%E2%80%93Fraenkel_set_theory?oldid=728821574 *Contributors:* AxelBoldt, Matthew Woodcraft, Zundark, Tarquin, Toby Bartels, Dwheeler, Patrick, Michael Hardy, MartinHarper, Bcrowell, Chinju, Haakon, Habj, Tim Retout, Schneelocke, Charles Matthews, Dcoetzee, Dysprosia, Greenrd, Hyacinth, VeryVerily, Fibonacci, JohnH~enwiki, Aleph4, Rursus, Tea2min, Giftlite, Smjg, Dratman, CyborgTosser, Mellum, Jorend, Ajgorhoe, Tarantoga~enwiki, David Sneek, Vsmith, Bender235, Elwikipedista~enwiki, Peter M Gerdes, Nortexoid, Obradovic Goran, Msh210, Suruena, TXlogic, Gible, Oleg Alexandrov, Joriki, OwenX, Drostie, Ma Baker, Hdante, Esben~enwiki, Dionyziz, Qwertyus, MarSch, Salix alba, R.e.b., STarry, Chobot, Karch, YurikBot, Hairy Dude, Michael Slone, Pi Delport, Ksnortum, Ogai, Trovatore, Twin Bird, Expensivehat, Insipid, Jpbowen, Crasshopper, Wknight94, Arthur Rubin, Josh3580, Banus, Otto ter Haar, Schizobullet, A bit iffy, SmackBot, Fulldecent, Mhss, Darth Panda, Foxjwill, Tsca.bot, Miguel1626, TKD, Allan McInnes, Grover cleveland, Acepectif, Jon Awbrey, Meni Rosenfeld, Stefano85, Vina-iwbot~enwiki, Noegenesis, Rainwarrior, Dicklyon, Tophtucker, JRSpriggs, CRGreathouse, CBM, Myasuda, Gregbard, Awmorp, Thijs!bot, Whooooooknows, Odoncaoa, Jirka6, VictorAnyakin, JAnDbot, Quentar~enwiki, Giler, Mathfreq, Omicron18, JustinRosenstein, Diroth, The Real Marauder, Numbo3, Fruits Monster, Trumpet marietta 45750, Policron, JavierMC, The enemies of god, Alan U. Kennington, Crisperdue, Pasixxxx, Magmi, Bistromathic, Henry Delforn (old), Yoda of Borg, Jjepfl, C xong, JP.Martin-Flatin, Alexbot, Iohannes Animosus, Palnot, Marc van Leeuwen, Addbot, Matěj Grabovský, Yobot, AnomieBOT, Materialscientist, La comadreja, Control.valve, VladimirReshetnikov, Nicolas Perrault III, Andrewjameskirk, NSH002, Tkuvho, Zdorovo, ClueBot NG, Chetrasho, Wcherowi, Snotbot, Helpful Pixie Bot, BG19bot, Brad7777, Daysrr, Khazar2, Jochen Burghardt, Mark viking, Trackteur, Magriteappleface, Pivotcoptr, Baking Soda and Anonymous: 126

- **Presburger arithmetic** *Source:* https://en.wikipedia.org/wiki/Presburger_arithmetic?oldid=729403846 *Contributors:* Damian Yerrick, AxelBoldt, LC~enwiki, 0, Taw, Andre Engels, XJaM, Vkuncak, Michael Hardy, Chinju, Zeno Gantner, Paddu, Ajk, Tim Retout, Wik, AnonMoos, David.Monniaux, Ruakh, Giftlite, Gdr, Mike Rosoft, Smimram, Guanabot, Pavel Vozenilek, Petrus~enwiki, Ben Standeven, Pmetzger, Spayrard, EmilJ, HasharBot~enwiki, Zenosparadox, RJFJR, Oleg Alexandrov, Graham87, BD2412, Rjwilmsi, R.e.b., Trovatore, Jpbowen, JCSantos, Janm67, Clements, Lambiam, Nagle, Mets501, Zero sharp, ILikeThings, CRGreathouse, CBM, Myasuda, Gregbard, Erxnmedia, Thenub314, Nyq, DWIII, Rogator, Sapphic, VVVBot, C. lorenz, Addbot, DOI bot, AnnaFrance, Derekoppen, Matěj Grabovský, Yobot, Miym, Confront, Citation bot 1, Arthur MILCHIOR, OriumX, Erwinrcat, EmausBot, ZéroBot, Quondum, Dagko, George Makepeace, Helpful Pixie Bot, BG19bot, BattyBot, Jochen Burghardt, There is a T101 in your kitchen, Gasole and Anonymous: 46
- **Lambda calculus** *Source:* https://en.wikipedia.org/wiki/Lambda_calculus?oldid=730337749 *Contributors:* AxelBoldt, Tobias Hoevekamp, General Wesc, Dreamyshade, Derek Ross, Vulture, LC~enwiki, CYD, Bryan Derksen, The Anome, Jan Hidders, Andre Engels, Arvindn, Fubar Obfusco, Spiff~enwiki, Michael Hardy, Dominus, GTBacchus, Eric119, Minesweeper, Cyp, Nanshu, Glenn, AugPi, Tim Retout, Smack, Charles Matthews, Timwi, Dysprosia, Malcohol, Markhurd, Furrykef, Populus, Shizhao, Kwantus, Joy, MH~enwiki, Robbot, Lloyd~enwiki, Tomchiukc, Kowey, MathMartin, Jleedev, Tea2min, David Gerard, Ancheta Wis, Gploc, Centrx, Giftlite, Paul Richter, MathKnight, Herbee, Everyking, Jonabbey, Dratman, Esap, Jorend, Jason Quinn, Neilc, Nomeata, Vadmium, PenguiN42, James Crippen, Sarex, Urhixidur, Bbpen, Karl Dickman, Corti, Mormegil, Rfl, Pyrop, Zaheen, Rich Farmbrough, Guanabot, Leibniz, FiP, ArnoldReinhold, Smyth, Slipstream, Bender235, ZeroOne, Elwikipedista~enwiki, MBisanz, Chalst, Tjic, Idmillington, Blonkm, John Vandenberg, Cmdrjameson, Jjk, Lance Williams, Kensai, Chbarts, Matt McIrvin, Tromp, Zygmunt lozinski, Hooperbloob, Diego Moya, Keenan Pepper, Sligocki, Evil Monkey, Tony Sidaway, Gpvos, TenOfAllTrades, Bfreis, LunaticFringe, Oleg Alexandrov, Siafu, Thryduulf, Pekinensis, Wedesoft, Linas, Jyavner, LOL, MattGiuca, Ruud Koot, Gruu, Apokrif, GregorB, Palica, Marudubshinki, BD2412, Qwertyus, Jacob Finn, Rjwilmsi, MarSch, Venullian, Vegaswikian, Husky, Titoxd, VKokielov, EvanSeeds, John Baez, Mathbot, Grd-Scarabe~enwiki, Lexspoon, Makkuro, StevenDaryl, Natkuhn, Vonkje, Chobot, YurikBot, Wavelength, Hairy Dude, D.keenan, RussBot, Michael Slone, Koffieyahoo, Jtbandes, BlakeStone, CarlHewitt, Thunderforge, R.e.s., Joel7687, Kliph, Jpbowen, JulesH, BeastRHIT, Bota47, Hakeem.gadi, CLW, Wknight94, Ms2ger, Saric, Liyang, Thnidu, Cedar101, Donhalcon, JLaTondre, GrinBot~enwiki, Ffangs, SmackBot, Pintman, TobyK, Ariovistus, Atomota, BiT, Sam Pointon, Mcld, Hmains, PJTraill, Serhei Makarov, Chris the speller, Optikos, Theone256, Jyossarian, Jimmyzimms, Mcaruso, Javalenok, Dfletter, Khukri, Derek R Bullamore, Iridescence, JamieVicary, Daniel.Cardenas, OlegAndreev, Lambiam, Antonielly, Physis, Mets501, Kripkenstein, Iridescent, JMK, Gabn1, Zero sharp, Tawkerbot2, Nerfer, Wafulz, CBM, Nczempin, Neelix, Sanspeur, Gregbard, Cydebot, Krauss, Illpoint, Sam Staton, Blaisorblade, Julian Mendez, Msreeharsha, Torc2, Nowhere man, Roccorossi, Bsmntbombdood, Morgaladh, Kirk Hilliard, Wikid77, Henk Barendregt, N5iln, Pjvpjv, Hjoab, Escarbot, KrakatoaKatie, AntiVandalBot, Whiteknox, Gioto, Kba, MBParker, Dougher, JAnDbot, The Transhumanist, Rearete, Workaphobia, Magioladitis, Abcarter, Fuchsias, Xb2u7Zjzc32, TehKeg, A3nm, David Eppstein, Shredderyin, Tigrisek, Gwern, John Millikin, R'n'B, GoatGuy, Daniel5Ko, CeilingCrash, Raise exception, NewEnglandYankee, Policron, ScottBurson, Largoplazo, Potatoswatter, Ultra two, Adam1729, Pborenstein, The enemies of god, Meiskam, DPMulligan, LokiClock, TXiKiBoT, Mattc58, Marcosaedro, Shades79, Softtest123, TelecomNut, Jesin, GlassFET, Aeris-chan, Radagast3, AmigoNico, SieBot, Equilibrioception, Gerakibot, Toddst1, Evaluist, Dddenton, Ctxppc, DancingPhilosopher, Valeria.depaiva, Dcattell, Longratana, Classicalecon, ClueBot, The Thing That Should Not Be, Dented42, DavisSta, EoGuy, Maniac18, Nonlinear149, HowardBGolden, Adrianwn, Ironrange, Hyh1048576, UKoch, Cinayakoshka, Alexbot, Pmronchi, Numl0k, Rhododendrites, Hans Adler, Alexey Muranov, Awegmann, Dwiddows, AndreasBWagner, Afarnen, Hugo Herbelin, StevenDH, Qwfp, Joswig, Brianpeiris, Dreblen, Kizeral, Addbot, Trueyoutrueme, Some jerk on the Internet, Landon1980, Punto7, AnnaFrance, Roaryk, Numbo3-bot, Lightbot, Jarble, Pedzsan, Luckas-bot, Yobot, Ht686rg90, Pcap, AnomieBOT, Enisbayramoglu, Royote, Karun.mahajan, Пика Пика, Citation bot, Charlieegan3, Xqbot, Jebdm, YBG, Jonas AGX, Noamz, Stratocracy, CLeJ37, Aaditya 7, Bstarynk, Adamuu, FrescoBot, Util PM, ComputScientist, Artem M. Pelenitsyn, Sae1962, Demosthenes2k8, Umawera, Kallocain, Citation bot 1, Pinethicket, Sherjilozair, WLoku, WaddSpoiley, CodeBlock, RedBot, WuTheFWasThat, Burritoburritoburrito, Lueckless, Lotje, Holoed, Edinwiki, Mrout, WillNess, LoStrangolatore, Fappah, Vcunat, FalseAxiom, Bgeron, Wgunther, KHamsun, Grondilu, ZéroBot, ArachanoxReal, D.Lazard, Future ahead, Stanmanish, ResearchRave, ClueBot NG, Piast93, Ghartshaw, Frietjes, Thepigdog, Nullzero, Helpful Pixie Bot, J.Dong820, Wbm1058, BG19bot, Lawandeconomics1, Halfb1t, TypesGuy1234, Wolfgang42, Glacialfox, ChrisGualtieri, Pintoch, Sean htx, Dtm1234, Splendor78, Paul1andrews, TheKing44, Mgns51, Jochen Burghardt, Dschslava, Pdecalculus, Melonkelon, BeastCCT, Julia Abril, Glassbreaker, EXist73, MergerDude, Paul2520, Aubreybardo, Ordnungswidrig, Jayaguru-Shishya, OMPIRE, Zeiimer, Axiarchist, Aasasd, Wombur, Mikhdm, Shubhamsoumit, Spliff Joint Blunt, Iglpdc, Matrix1919, WikipediaIsAwesome9999, Oknaye, Paxwell, Olrik 113, Camaiocco, WallyWinker and Anonymous: 445
- **Gödel's incompleteness theorems** *Source:* https://en.wikipedia.org/wiki/G%C3%B6del'{}s_incompleteness_theorems?oldid=729773500 *Contributors:* AxelBoldt, Joao, Chenyu, LC~enwiki, Lee Daniel Crocker, Mav, Bryan Derksen, The Anome, Tarquin, Jan Hidders, Andre Engels, Danny, MadSurgeon, SimonP, Camembert, Genneth, Olivier, Michael Hardy, Tim Starling, Kwertii, Dominus, Lousyd, MartinHarper, Wapcaplet, Chinju, TakuyaMurata, GTBacchus, Eric119, CesarB, HarmonicSphere, ThirdParty, Ootachi, Suisui, Den fjättrade ankan~enwiki, Mark Foskey, Александър, BuzzB, AugPi, Tim Retout, Rotem Dan, Evercat, Madir, Gamma~enwiki, Rzach, Charles Matthews, Timwi, Dcoetzee, Reddi, Dysprosia, Wikid, Doradus, Markhurd, Lfwlfw, Hyacinth, Bevo, Joseaperez, .mau., Gakrivas, Pakaran, Ldo, David.Monniaux, Dmytro, Owen, Aleph4, Altenmann, Gandalf61, MathMartin, Bethenco, Rasmus Faber, Bkell, Mjscud, Aetheling, Ruakh, Tea2min, Jimpaz, Ancheta Wis, Tosha, Matt Gies, Giftlite, Lethe, Lupin, Fropuff, Mellum, Waltpohl, Geoffroy~enwiki, Wikiwikifast, Andris, Bovlb, Sundar, Prosfilaes, Siroxo, Khalid hassani, C17GMaster, Neilc, Utcursch, Andycjp, Sigfpe, Toytoy, Gdr, SarekOfVulcan, Gdm, Lightst, Antandrus, Beland, Amoss, APH, DragonflySixtyseven, Mike Storm, Elroch, Sam Hocevar, Asbestos, Karl Dickman, Gazpacho, D6, Sysy, 4pq1injbok, Guanabot, FT2, Cacycle, Smyth, Francis Davey, Maksym Ye., Paul August, Bender235, Chalst, Crisófilax, EmilJ, Lance Williams, DG~enwiki, AshtonBenson, Jumbuck, Keenan Pepper, Hu, Wtmitchell, Simplebrain, Cal 1234, GabrielF, Gene Nygaard, MIT Trekkie, TXlogic, Oleg Alexandrov, Revived, Billhpike, OwenX, Rodrigo Rocha~enwiki, Pol098, Ruud Koot, Apokrif, Frungi, Waldir, Marudubshinki, MACherian, Mandarax, Graham87, BD2412, Qwertyus, Rjwilmsi, Tim!, Kinu, WoodenTaco, RCSB, Pasky, R.e.b., Magidin, FlaBot, VKokielov, Docbug, Mathbot, Jrtayloriv, Celendin, Sperxios, Xelloss~enwiki, Eric.dane~enwiki, Chobot, Bgwhite, YurikBot, Wavelength, Hairy Dude, Dmharvey, Arado, Rcaetano, Me and, Ksnortum, JabberWok, KSmrq, Bergsten, SpuriousQ, Thoreaulylazy, Vibritannia, Ytcracker, Aatu, Trovatore, Długosz, JoeBruno, Dogcow, Nick, Anetode, Philosofool, Guruparan18, Hakeem.gadi, Liyang, Arthur Rubin, Curpsbot-unicodify, GrinBot~enwiki, Finell, SolarMcPanel, SmackBot, RDBury, Selfworm, BeteNoir, InverseHypercube, Melchoir, SaxTeacher, Pokipsy76, Mandelum, Biedermann, Brick Thrower, Srnec, The Rhymesmith, Chris the speller, MagnusW, Irving Anellis, RoyArne, MalafayaBot, Kostmo, Jdthood, Charles Moss, Grover cleveland, AndySimpson, Trifon Triantafillidis, John wesley, Kid A~enwiki, LoveMonkey, Rhkramer, Byelf2007, Zchenyu, Lambiam, Wvbailey, Michael L. Hall, Evildictaitor, Illythr, Mets501, ASKingquestions, Sgutkind, Dr.K., Nbhatla, Quaeler, Dan Gluck, Jason.grossman, Dreftymac, Joseph Solis in Australia, Zero sharp, Wfructose, Matthew Kornya, JRSpriggs, CRGreathouse, Geremia, Diegueins, CBM, Nicolaennio, Gregbard, Nilfanion, Sopoforic, Cydebot, Pce3@ij.net, Steel, Peterdjones, TicketMan, Mon4, Blaisorblade, Michael C

Price, DumbBOT, Robertinventor, Morgaladh, Clickheretologin, Thijs!bot, Headbomb, Marek69, Tamalet, Towopedia, Turkeyphant, Kborer, Stan the fisher, Joegoodbud, Gioto, Blue Tie, Ad88110, NBeale, Husond, Avaya1, Gavia immer, JamesBWatson, Renosecond, Tedickey, Baccyak4H, K95, David Eppstein, Exiledone, Pavel Jelínek, Infovarius, Franp9am, Eliko, Abecedare, Fredeaker, Warut, CeilingCrash, DadaNeem, Policron, Mad7777, DavidCBryant, Merzul, Izno, Quux0r, Germanium, Alan U. Kennington, VolkovBot, DDSaeger, AlnoktaBOT, TXiKiBoT, Sacramentis, Lou2261, CaptinJohn, Broadbot, David in DC, Lifeisfun0007, Latulla, Popopp, Artem-S-Tashkinov, Gbawden, Davebuehler, YohanN7, SieBot, Simplifier, Iamthedeus, Gerakibot, Noaqiyeum, Likebox, Flyer22 Reborn, Scouto2, OKBot, Valeria.depaiva, D14C050, IsleLaMotte, Anchor Link Bot, S2000magician, Tesi1700, CBM2, ReluctantPhilosopher, Beeblebrox, ClueBot, Admiral Norton, DFRussia, Pi zero, Juustensson, Razimantv, Boing! said Zebedee, Niceguyedc, Libett, Ajoykt, Ademh, Byates5637, Bender2k14, Sun Creator, Coinmanj, AmirOnWiki, Tnxman307, Snacks, Hans Adler, Muro Bot, Darkicebot, Palnot, Gerhardvalentin, TravisAF, Xamce, Addbot, Yousou, Harttwood47, RPHv, Mpholmes, EjsBot, Vishnava, Download, LaaknorBot, ב דניאל., Legobot, Luckas-bot, Yobot, Ht686rg90, TaBOT-zerem, Nallimbot, Third Merlin, FeydHuxtable, AnomieBOT, 9258fahsflkh917fas, Materialscientist, Citation bot, Twiceuponatime, Markworthen, LilHelpa, Obersachsebot, Xqbot, Psyoptix, False vacuum, VladimirReshetnikov, Shaun.mk, Kristjan.Jonasson, Andrewjameskirk, Mark Renier, Mfwitten, Citation bot 1, Þjóðólfr, Sptzimas, Tkuvho, Aunin, MarcelB612, MPeterHenry, Standardfact, H.ehsaan, RjwilmsiBot, Fictionalist, Kozation, EmausBot, Nameguy101, PBS-AWB, Bijuro, AvicAWB, Negyek, Dominique.devriese, Herk1955, Llightex, CasualUser1729, ClueBot NG, Kevin Gorman, Helpful Pixie Bot, Wbm1058, Lowercase sigmabot, BG19bot, Slippingspy, Modelpractice, Cyberpower678, Minsbot, Marktoiii0, Kiewbra, Cyberbot II, Deltahedron, Fernandodelucia, Harri jensen, LFOlsnes-Lea, Paul1andrews, Mark viking, Lcpaulson, EvergreenFir, The Herald, Dinadineke, Eng.alireda, 22merlin, Leegrc, Zeus000000, Latinosopher, Zppix, ScrapIronIV, Oeoi, Wen1now, Baking Soda and Anonymous: 378

- **Proof sketch for Gödel's first incompleteness theorem** *Source:* https://en.wikipedia.org/wiki/Proof_sketch_for_G%C3%B6del'{}s_first_incompleteness_theorem?oldid=725799721 *Contributors:* Michael Hardy, Giftlite, FT2, Ben Standeven, Woohookitty, Mangojuice, BD2412, Rjwilmsi, Salix alba, Destin, Zvika, JF Manning, Colonies Chris, Jgrahamc, Zchenyu, Dan Gluck, CBM, Gregbard, Cydebot, DumbBOT, PamD, Heysan, R'n'B, Palaeovia, YohanN7, Nbanic, Philosophy.dude, Hans Adler, Addbot, Yobot, AnomieBOT, EmausBot, Lowercase sigmabot, BG19bot, Kiewbra, CarrieVS, Hobeewahn, Faizan, Baking Soda and Anonymous: 26

7.9.2 Images

- **File:Codomain2_A_B.SVG** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/d1/Codomain2_A_B.SVG *License:* Public domain *Contributors:* Own work (Original text: *I created this work entirely by myself.*) *Original artist:* Damien Karras (talk)
- **File:Edit-clear.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/f/f2/Edit-clear.svg *License:* Public domain *Contributors:* The *Tango! Desktop Project*. *Original artist:*

  The people from the Tango! project. And according to the meta-data in the file, specifically: "Andreas Nilsson, and Jakub Steiner (although minimally)."
- **File:Logic_portal.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/7/7c/Logic_portal.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Watchduck (a.k.a. Tilman Piesk)
- **File:Nuvola_apps_edu_mathematics_blue-p.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/3/3e/Nuvola_apps_edu_mathematics_blue-p.svg *License:* GPL *Contributors:* Derivative work from Image:Nuvola apps edu mathematics.png and Image:Nuvola apps edu mathematics-p.svg *Original artist:* David Vignoni (original icon); Flamurai (SVG convertion); bayo (color)
- **File:Predicate_logic;_2_variables;_example_matrix_a(12).svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/53/Predicate_logic%3B_2_variables%3B_example_matrix_a%2812%29.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Predicate_logic;_2_variables;_example_matrix_a12.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/55/Predicate_logic%3B_2_variables%3B_example_matrix_a12.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Predicate_logic;_2_variables;_example_matrix_a1e2_nodiag.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/c/ce/Predicate_logic%3B_2_variables%3B_example_matrix_a1e2_nodiag.svg *License:* CC BY-SA 3.0 *Contributors:* Own work, based on File:Predicate_logic;_2_variables;_example_matrix_a1e2.svg (didn't clean-up the mess in the svg source before modifying) *Original artist:* Jochen Burghardt
- **File:Predicate_logic;_2_variables;_example_matrix_a2e1.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/62/Predicate_logic%3B_2_variables%3B_example_matrix_a2e1.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Predicate_logic;_2_variables;_example_matrix_e(12).svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/f0/Predicate_logic%3B_2_variables%3B_example_matrix_e%2812%29.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Predicate_logic;_2_variables;_example_matrix_e12.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/5d/Predicate_logic%3B_2_variables%3B_example_matrix_e12.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Predicate_logic;_2_variables;_example_matrix_e1a2.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/0/02/Predicate_logic%3B_2_variables%3B_example_matrix_e1a2.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Predicate_logic;_2_variables;_example_matrix_e2a1.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/3/3a/Predicate_logic%3B_2_variables%3B_example_matrix_e2a1.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Predicate_logic;_2_variables;_implications.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/5b/Predicate_logic%3B_2_variables%3B_implications.svg *License:* Public domain *Contributors:* Own work *Original artist:* Watchduck (a.k.a. Tilman Piesk)
- **File:Prop-tableau-4.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/21/Prop-tableau-4.svg *License:* CC-BY-SA-3.0 *Contributors:* Transferred from en.wikipedia to Commons by Piquart using CommonsHelper. *Original artist:* Tizio at English Wikipedia
- **File:Question_book-new.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/9/99/Question_book-new.svg *License:* Cc-by-sa-3.0 *Contributors:*

  Created from scratch in Adobe Illustrator. Based on Image:Question book.png created by User:Equazcion *Original artist:*

  Tkgd2007
- **File:Text_document_with_red_question_mark.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/a4/Text_document_with_red_question_mark.svg *License:* Public domain *Contributors:* Created by bdesham with Inkscape; based upon Text-x-generic.svg from the Tango project. *Original artist:* Benjamin D. Esham (bdesham)

- **File:Venn_A_intersect_B.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/6d/Venn_A_intersect_B.svg *License:* Public domain *Contributors:* Own work *Original artist:* Cepheus

7.9.3 Content license