GÖDEL'S THEOREM SIMPLIFIED

SUMMARY AND REVISIONS

Kevin Carmody
1000 University Manor Drive, Apt. 34
Fairfield, Iowa 52556
i@kevincarmody.com

Third edition
27 June 2017

Third edition 27 June 2017
Second edition 28 March 2015
First edition 4 October 2011

CONTENTS

INTRODUCTION

Gödel's Theorem

Gödel's Theorem, more precisely Gödel's First Incompleteness Theorem, proves that any consistent, sufficiently rich axiomatic system of ordinary arithmetic contains statements that can be neither proved nor disproved. This theorem shatters the hope, which many mathematicians had harbored, that a finite system of axioms and logic could decide all mathematical propositions. Since the axiomatic approach of mathematics is widely believed to be the only way to obtain certain knowledge, and since Gödel's Theorem places a significant theoretical limit on the potential of this system to prove everything, the theorem has had a profound impact on the philosophy of mathematics, and on philosophy in general.

The mathematician and logician Kurt Gödel (1906–1978) published his famous theorem in [G31a], but this reference is of little direct value, since the paper is one of the most famous in the history of mathematics, and copies of the journal issue it was published in have long disappeared into the hands of souvenir hunters. Fortunately, the article has been republished and translated many times, including an approved English translation [G31b].

Gensler's book on Gödel's theorem

Gödel's Theorem is technically difficult. Gödel's original article was written for his fellow researchers. It assumes much background material that was known to researchers of the time, and it uses a now-obsolete notation. Expositions of Gödel's Theorem since that time have improved the situation by providing background material, alternative explanations, and a current notation, but the topic is still usually reserved for graduate study in symbolic logic.

Fr. Harry J. Gensler in his *Gödel's Theorem Simplified* [Ge84] aims to make Gödel's theorem accessible to a general undergraduate audience, without sacrificing rigor. Gensler separates elementary material from technically advanced material and makes the advanced material optional. He includes much background material, proceeds in very easy steps when presenting the elementary material, uses a formula numbering scheme that is simpler than Gödel's, and aims to write a complete set of formulas to construct the "Gödel formula", the existence of which proves Gödel's Theorem.

In the author's opinion, Gensler has admirably succeeded in most of these aims. The elementary material is very accessibile, and this alone makes it an excellect exposition of Gödel's Theorem. However, there is a serious technical flaw, discussed in the next section.

The need for revisions

Chapter V of Gensler's book presents an overview of the construction of the Gödel formula, and Chapter VII presents a detailed set of formulas that are used to construct the Gödel formula. Gensler presents a very detailed and apparently rigorous series of formulas which he says can be used to generate the Gödel formula. He rigorously constructs the precursor to the Gödel formula, which he calls the "father" of the Gödel formula. Yet, at the very end, he cuts the process short—he does not generate the Gödel formula itself, the "son" of the father formula. Moreover, the formulas he constructs do not allow one to even prove that the Gödel formula exists, much less construct it. This is a serious matter, since the validity of Gödel's Theorem completely depends upon the existence of the Gödel formula, which Gödel originally proved by actually constructing it. Almost all of Chapter VII would have to be deeply rewritten to adequately construct the Gödel formula using Gensler's approach. Since the rest of Gensler's treatment of Gödel's Theorem is so admirable, the author has undertaken this task below.

In Chapter I (p. 1) Gensler paraphrases Gödel's Theorem as ***Arithmetic cannot be reduced to any axiomatic system*** then carefully defines these terms (p. 2):

- ***Arithmetic*** is a system that includes adding and multiplying positive whole numbers, variables, and simple logical operations.
- ***Axiomatic system*** includes axioms and rules of logical inference.
- ***Reduction*** means that every formula that is true in the language of the system can be proved, but no false formula can be proved.

Gensler's concept of reduction is different from Gödel's concept of completeness, as Gensler briefly explains in Chapter VI (p. 64). A system is complete if every statement can be either proved or disproved, i.e. for every statement, either it or its negation can be proved. Reduction is weaker than completeness but still conveys the essence of the theorem. This is examined in broader detail below.

Gensler's book does not consider recursive functions or set theoretical constructs such as transfinite numbers, which are addressed in Gödel's paper. We will also not touch on these topics, since they are not central to the theme of Gödel's Theorem.

Spelling of “Gödel” and other typographical issues

Gödel’s Theorem Simplified was published in 1984 using an early word processing system that had only a typewriter font. The book consistently omits the umlaut (two dots) above the “o” in “Gödel”, spelling it “Godel”, even in the title. German orthography requires, however, that umlauts not simply be omitted from German words. Whenever it is not possible to type “ä”, “ö”, or “ü”, they should be written “ae”, “oe”, or “ue”. This gives two proper forms of the name of the discoverer of the Theorem, “Gödel” and “Goedel”.

The typewriter font used in Gensler’s book also means that it does not use certain now-standard mathematical symbols. For instance, the book uses “v”, “-”, and “E” instead of $\vee$, $\neg$, and $\exists$. This document uses an updated version of the notation of the book.

These deficiences are rectified in Gensler’s later books, which use modern word processing systems. One such book is Gensler’s *Introduction to Logic* [Ge10], which on pages 345–350 contains a condensed version of the main argument in *Gödel’s Theorem Simplified* in standard mathematical notation. This later book of Gensler cites his earlier book as *Gödel’s Theorem Simplified*, with the correct umlaut in “Gödel”.

SYSTEMS AND THEIR LANGUAGES

Systems

Gödel's Theorem applies to a formal mathematical system, which comprises:

- a language for expressing mathematical terms, statements, and proofs
- a set of axioms
- a set of inference rules, which specify how one or two statements can be transformed into another statement
- the restriction of mathematical statements to positive whole numbers only.

To help put Gödel's Theorem in context, Gensler considers several alternatives to the standard mathematical system, and shows how analogs of Gödel's Theorem holds in some of these systems and not in others. The following is a brief summary of these systems, including a system not on Gensler's list:

- System C: the standard mathematical system in which Gödel's Theorem holds
- System B: a system nearly identical to System C, in which Gödel's Theorem does not hold
- System A: a system somewhat simpler than System B, in which Gödel's Theorem does not hold
- System E: a very simple system, in which Gödel's Theorem holds
- System F: an even simpler system than System E, in which Gödel's Theorem does not nold (and which is not in Gensler's book)

Language of the systems

Each system uses a carefully constructed formal language to represent mathematical terms, statements, and proofs that are allowed in that system. The exact language varies from one system to another. For reasons that will become clear later, it is important to keep the number of symbols in each language to a minimum.

A ***string*** in any given language is a sequence of symbols that are used in that language. A string is meaningful only if it follows a set of strict rules called ***grammar rules***.

In each system, numbers are handled as follows:

- The only numbers that the language can represent are positive integers.
- Numbers are represented through repetitions of a stroke symbol /: 1 is represented by /, 2 by //, 5 by /////, 10 by //////////, etc.
- The operations allowed on these numbers vary from one system to another, but never include more than addition, multiplication, and exponentiation, using the symbols +, $*$, and $**$.
- The only relation that is used is equality. A statement of equality is called a ***formula***. // = // is a true formula, and /// = // is a false formula.
- Variable names are constructed by repeating one symbol, n. Possible variable names are thus n, nn, nnn, $nnnn$, etc.

Some systems include logical operators:

- A star ($*$) between formulas means logical conjunction. ($*$ between numbers means multiplication.)
- The symbol ($\neg$) before a formula means logical negation.
- Parentheses around a variable name mean a quantifier. The notation $(n)\ n = n$, for example, means that $n = n$ for all n. The quantifier (n) is elsewhere often denoted $(\forall n)$, but the symbol $\forall$ is omitted from the Gensler systems.

A comma (,) is used to join formulas into sequences. A sequence forms a proof when each formula in the sequence is either an instance of an axiom or the result of applying an inference rule to one or more preceding formulas. The last formula in a proof sequence is a ***theorem***, i.e. the sequence forms a proof of the theorem. For instance, in System C, the sequence `/// = ///, // + / = ///` is a proof of the theorem 2 + 1 = 3:

$/// = ///$ is an instance an axiom which states that any number is equal to itself, and $// + / = ///$ is the result of applying to $/// = ///$ an inference rule which states that $h + /$ is interchangable with $h/$ for any number h.

The language of each system can be extended by other symbols which are shortcuts into the formal language. For instance, logical disjunction is introduced by defining $a \vee b$ as $\neg(\neg a * \neg b)$, where a and b are formulas. Similarly, the existential quantifier notation $(\exists u)\ p(u)$, where u is a variable and $p(u)$ is a formula involving u, is defined as $\neg(u)\ \neg p(u)$.

SYSTEMS E AND F AND THEIR INTERPRETATIONS

System F

Systems E and F are so important to the proper understanding of Gödel's Theorem that we will examine them here in detail. We start with the simplest system, System F.

There are only two symbols in the language:

$$/ \qquad =$$

There are two grammar rules:

1. A string consisting of one or more / is a ***numeral***.
2. A string of the form $x = y$, where x and y are numerals, is a ***formula***.

There is only one axiom: $x = x$ for any numeral x. Instances of this axiom include / = /, // = //, etc.

There are no inference rules. Inference rules occur in Systems A and B and in System C.

It is crucial to understand the difference between the axiom and the formula $(x)\ x = x$. The latter formula is not possible in this system, because the language of this system does not include quantifiers. The axiom $x = x$ is actually an ***axiom schema***: it stands for any of an infinite number of formulas of this form, each of which is called an ***instance*** of the axiom. In other words, the axiom allows us to state that any one number is equal to itself, but it does not allow us to state that *all* numbers are equal to themselves.

Therefore:

- Only formulas of the form $x = x$ are true, and these are all instances of the axiom.
- Since there are no inference rules, proofs can consist of only one formula, which must be an instance of the axiom, namely a formula of the form $x = x$ for some numeral x.
- These formulas are the only possible theorems.

- This system is ***complete*** (all theorems expressible in the language of the system can be proved within the system) and ***sound*** (all theorems that the system proves are true).

- Gödel's Theorem does *not* hold in this system.

System E

System E adds quantifiers to System F.

There are five symbols in the language:

$$/ \quad (\quad) \quad = \quad n$$

where the interpretation of ( ) is only for quantifiers, not for grouping.

There is only one axiom, which is identical to the axiom of System F: $x = x$ for any numeral x.

There are four grammar rules:

1. A string consisting of one or more / is a ***numeral*** and a ***term***.
2. A string consisting of one or more ***n*** is a ***variable*** and a ***term***.
3. If x and y are terms, then a string in the form $x = y$ is a ***formula***.
4. If v is a variable and f is a formula, then a string in the form $(v)\ f$ is a ***formula***.

As in System F, there are no inference rules.

The following types of formula are possible:

- $x = y$, where x and y are numerals

- $(n)\ n = n$

- $(n)\ n = x$ for some numeral x

- $(n)\ x = n$ for some numeral x

Of these, the following are true:

- $x = x$

- $(n)\ n = n$

As in System F, in System E:

- There are no inference rules, so proofs can consist of only one formula, which must be an instance of the axiom, namely a formula of the form $x = x$ for some numeral x.

- These formulas are the only possible theorems.

Unlike System F, in System E:

- The language allows us to formulate the obviously true formula $(n)\ n = n$, but it is not a theorem. It cannot be proved from the axiom. The axiom can prove any individual instance of this formula, but it cannot prove the general formula.

- Gödel's Theorem *does* hold in this system.

From this consideration of Systems E and F, we see that a system can easily be made irreducible by adding symbols to the language that are not mentioned in the axioms. Reduction is the exception for axiomatic systems, not the rule.

Interpretations of Systems E and F

A comparison of Systems E and F shows that the language of a system of arithmetic may allow the formulation of true formulas which the axioms are not able to prove. Now we consider the role of interpretation to see that the truth of formulas in any system depends on the context in which the language of the system is used.

The consideration of Systems E and F above stated that $(n)\ n = n$ is obviously true, but in actuality it is possible to give alternate interpretations of the language of these systems in which this formula is not true. Below are a few possible interpretations.

Interpretation 1: The standard interpretation. Quantifiers range only over the positive integers, and $x = y$ has its usual meaning. $(n)\ n = n$ is obviously true in this interpretation.

Interpretation 2: Quantifiers range over all integers, including positive, negative, and zero. $x = y$ means that x and y are both positive and equal. The axioms of systems E and F are true, because they correctly equate positive integers to themselves, but $(n)\ n = n$ is not true, because when n is negative, $(n)\ n = n$ is false.

Interpretation 3: Quantifiers range over positive integers only. $x = y$ means that x and y are both even and equal. Some instances of the axiom of Systems E and F are true, but others are false.

Interpretation 4: Quantifiers range over all integers. $x = y$ means that both x and y are negative and equal. There are values of n for which $(n)\ n = n$, but they are all negative. All instances of the axioms of systems E and F are false—in fact all possible formulas in these systems are false.

SYSTEMS A and B

Outline of Systems A and B

System A corresponds roughly to primary school arithmetic and is called the ***arithmetic of addition***. It includes addition of positive integers, but not subtraction, multiplication, division, exponentiation, variables, logical operators, or quantifiers. As we will see, Gödel's Theorem does not hold in this system.

System B corresponds roughly to grade school arithmetic and is called as ***lower arithmetic***. It includes addition, multiplication, and exponentiation of positive integers, but not subtraction, division, variables, logical operators, or quantifiers. We will see that Gödel's Theorem also does not hold in this system.

System C is the standard system in which Gödel's Theorem holds and is dealt with in the next chapter, System C.

System A

System A contains statements of equality, with addition as a numeric operator.

There are five symbols in the language:

$$/ \quad (\quad) \quad + \quad =$$

where the interpretation of ( ) is only for grouping, not for quantifiers.

There is only one axiom, which is identical to the axiom of Systems E and F: $x = x$ for any numeral x.

There are three grammar rules:

1. A string consisting of one or more / is a ***numeral*** and a ***term***.
2. If x and y are terms, then the string $(x + y)$ is a ***term***.
3. If x and y are terms, then the string $x = y$ is a ***formula***.

There are two inference rules:

1. For any terms x and y, the term $(x + /)$ can be exchanged with $x/$.

2. For any terms x and y, the term $(x + y/)$ can be exchanged with $(x/ + y)$.

Terms may include branching to any level of complexity, such as:

$$///$$
$$/ + //$$
$$// + /$$
$$/ + ////$$
$$/ + (/ + //)$$
$$(/ + (// + (/// + /) + //) + ///) + //$$

The following types of formula are possible:

- $u = v$, where u and v are numerals
- $u = x$ or $u = x$, where u is a numeral and x is a term that is not a numeral
- $x = y$, where x and y are terms that are not numerals

Of these, the following are true:

- $u = u$, where u is a numeral
- $u = x$ or $u = x$, where u is a numeral and x is a term that can be transformed into u by one or more applications of the inference rules
- $x = y$, where x and y are terms that can be transformed into the same numeral by one or more applications of the inference rules

Using the inference rules, the symbols +, (, and) can be eliminated from any term, leaving only / and =. This can be used to transform any formula into the form $u = v$. If u and v are the same numeral, then the transformed formula is an instance of the axiom. Under the standard interpretation of the symbols, such a formula is true. By reversing this process, any instance of the axiom can be transformed into a true formula.

Since this can be done for any formula expressible in the language of System A, any true formula that is expressible in the language of System A can be proved, and any formula that can be proved is true; no false formula can be proved.

System A is therefore ***reducible*** (every true theorem can be proved and every provable theorem is true). Gödel's Theorem does *not* hold in System A.

System B

System B adds multiplication and exponentiation to System A.

There are six symbols in the language:

$$/ \quad (\quad) \quad + \quad * \quad =$$

where the interpretation of ( ) is for grouping only, $*$ means multiplication, and $**$ means exponentiation.

There is only one axiom, which is identical to the axiom of Systems A, E, and F: $x = x$ for any numeral x.

There are five grammar rules:

1. A string consisting of one or more / is a ***numeral*** and a ***term***.
2. If x and y are terms, then the string $(x + y)$ is a ***term***.
3. If x and y are terms, then the string $(x * y)$ is a ***term***.
4. If x and y are terms, then the string $(x * *y)$ is a ***term***.
5. If x and y are terms, then the string $x = y$ is a ***formula***.

There are six inference rules:

1. For any terms x and y, the term $(x + /)$ can be exchanged with $x/$.
2. For any terms x and y, the term $(x + y/)$ can be exchanged with $(x/ + y)$.
3. For any term x, the term $(x * /)$ can be exchanged with x.
4. For any terms x and y, the term $(x * /y)$ can be exchanged with $((x * y + x)$.
5. For any term x, the term $(x * *1)$ can be exchanged with x.
6. For any terms x and y, the term $(x * */y)$ can be exchanged with $((x * *y) * x)$.

Some examples of true statements are:

$$/ + // = ///$$

$$// * /// = //////$$

$$// * */// = ////////$$

$$// * *(/ + //) = // * ////$$

$$/// * *// = ///// + ////$$

The same types of formula occur in System B that occur in System A. The inference rules of System B allow the symbols +, $*$, (, and) to be eliminated from any term, leaving only / and =. If the transformed formula is an instance of the axiom, then the formula is true under the standard interpretation. As in System A, reversing this process enables any instance of the axiom to be transformed into a true formula.

Since this can be done for any formula expressible in the language of System B, any true formula that is expressible in the language of System B can be proved, and any formula that can be proved is true; no false formula can be proved.

Therefore, System B, like System A, is ***reducible*** (every true theorem can be proved and every provable theorem is true). Gödel's Theorem does *not* hold in System B.

UNDERSTANDING SYSTEM C AND THE GÖDEL FORMULA

Outline of System C

System C is the standard mathematical system of arithmetic. It is also known as ***higher arithmetic*** since it corresponds to modern number theory. Like Systems A and B, it ranges only over the positive integers, and it includes addition, multiplication, and exponentiation. Unlike Systems A and B, it also includes logical operators, variables, and quantifiers. Gödel's Theorem *does* hold in System C, and this system is the main subject of Gensler's version of Gödel's Theorem.

Although "arithmetic" is the standard term for this system, it is often misleading, since the system includes variables and quantifiers, making it a form of algebra.

Systems A and B are weaker versions of System C. It was shown above that Gödel's Theorem does not hold in these systems.

Basic technique for proving Gödel's Theorem in System C

The basic strategy for proving Gödel's Theorem is to construct a statement which asserts its own unprovability. It essentially asserts *This statement is not provable*, but it is stated in the language of System C, where it is is called the ***Gödel formula*** and is denoted *G*.

The Gödel formula is similar to the Liar's Paradox, *This statement is false*. Gödel also proved an interesting result for the Liar's Paradox; see The Liar Paradox below. The Liar's Paradox can neither be true nor false—if it's true, then it's false, but if it's false, then it's true—and therefore such statements cause problems for standard logical systems.

However, the Gödel formula does not cause these problems, because it does not contradict itself—it only makes a statement about its provability. If the Gödel formula is true, then the language of System C contains an unprovable true statement, but if it's false, the language contains a provable false statement. In either case, System C is not reducible.

To construct the Gödel formula, we need:

- a language that encodes formulas and proofs in arithmetic
- a formula in this language that tests the provability of a formula
- a way to construct a formula that refers to itself

Language that encodes formulas and proofs in arithmetic

The language has 9 symbols, each of which has a numeric ID from 1 to 9.

Simple symbols

Symbol	*ID*	*Meaning*
/	1	One
+	2	Addition
$*$	3	Multiplication, logical conjunction
(	4	Left grouping, quantifier
)	5	Right grouping, quantifier
=	6	Equals
¬	7	Logical negation
n	8	Variable names
,	9	Formula separator

Compound symbols are pairs of simple symbols that would otherwise not be used together. Each compound symbol has a meaning separate from the meanings of the simple symbols and so has an alternate symbol.

Compound symbols

Symbol	*Alternate*	*Meaning*
$**$	↑	Exponentiation
$*$ =	IF	Conditional; see below
+ =	THEN	Conditional; see below
¬ =	ELSE	Conditional; see below
(=	FOR	Loop; see below
=)	UNTIL	Loop; see below
++	DO	Loop; see below

Conditional:

$$* = f + = x \neg = y$$

or

$$\text{IF } f \text{ THEN } x \text{ ELSE } y$$

is a conditional expression. For example, IF $//// = ///$ THEN $/$ ELSE $//$ can be paraphrased "If 4 = 3, then 1, else 2," and the expression as a whole has a value of 2.

Loop:

$$(= u = x =)\ f ++ y$$

or

$$\text{FOR } u = x \text{ UNTIL } f \text{ DO } y$$

is a loop expression. This is evaluated as follows:

1. Initially store the value x to the variable u.
2. Evaluate p, which is a formula whose truth depends on u.
3. If p is true, go to step 6.
4. If p is not true, evaluate y, which is a term whose value depends on u. Update the value of u to the value of y.
5. Go to step 2.
6. The value of the whole loop expression is the final value of u.

For example, $(n,\ = /,\ (nn)\neg(n + nn = 7),\ = n + /)$ can be paraphrased "For $n = 1$ until $(m)\ \neg(n + m = 7)$ do $n * 2$," or "Starting with $n = 1$, multiply n by 2 until n is greater than or equal to 7." The expression as a whole has a value of 8.

Any string of symbols, including any formula, also has an ID number, composed by stringing together the digits for each symbol into one number.

Formula	*Paraphrase*	*ID number*
$// + /// = /////$	$2 + 3 = 5$	112, 111, 611, 111
$(// + /) * // = //////$	$(2 + 1) \times 2 = 6$	4, 112, 153, 116, 111, 111
$(n)\ n + n = // * n$	For any n, $n + n = 2n$	48, 582, 861, 138
$(n)\ \neg(n + n = /// * n)$	There is no n such that $n + n = 3n$	485, 748, 286, 111, 385

A formula sequence can also be given a numeric ID, using the formula separator (,). A proof is a sequence in which each formula is either an instance of an axiom or the result of applying an inference rule to a preceding formula. Each formula in a proof sequence is a theorem. The following is a proof of the theorem 3 + 2 = 5.

Formula	*Justification*
///// = /////	1. Instance of axiom A1
//// + / = /////	2. Application of inference rule AR1 to formula 1
/// + // = /////	3. Application of inference rule AR2 to formula 2

We obtain the ID of the proof by stringing together the formulas with the formula separator.

Proof sequence:	///// = /////, //// + / = /////, /// + // = /////
ID number:	1,111,161,111,191,111,216,111,119,111,211,611,111

Constructing a formula that tests the provability of a formula

We now have a way to represent any formula and any proof as a number. Since the language can test for the existence of numbers satisfying certain properties, it can be used to test for the existence of the ID's of formulas and proofs, i.e. provability of formulas. From this point forward we work with numeric ID's of formulas and proofs rather than the formulas and proofs themselves.

The provability formula requires a series of definitions of shortcuts into the language. For instance, we will need logical disjunction, so for any two formula ID's p and q, we define $p \vee q$ as $\neg(\neg p * \neg q)$. The full set of definitions leads us to define a function $TH()$, where, for any formula ID p, $TH(p)$ means that there exists a proof for the formula which p represents.

Constructing a formula that refers to itself

We obtain a self-referential formula through a mechanism that Gensler calls ***father and son***.

First we define the ***son*** of a formula as the result of substituting the variable $\boldsymbol{n}$ in the formula with the numeric ID of the formula. For example, the ID of the formula $\boldsymbol{n} = \boldsymbol{n}$ is 868, and the son of this formula is another formula, 868 = 868. Actually, we have used shorthand notation for writing this second formula—if we wrote it out in full, it would be a string 1737 characters long: 868 /'s, then one =, then another 868 /'s. The ID of this formula would be a number 1737 digits long: 868 1's, then a 6, then another 868 1's.

The second step is to write a ***father*** formula F, which asserts that the son of formula number $\boldsymbol{n}$ is not a theorem.

Lastly, we write the son of this formula, which we denote G. To obtain G, we substitute the n in F with the ID for F. Then G asserts that the son of F is not a theorem. But the son of F is G—so G asserts that G is not provable. G is the Gödel formula. Its existence proves Gödel's Theorem.

FORMAL CONSTRUCTION OF THE GÖDEL FORMULA IN SYSTEM C

Background

Chapter V of Gensler's book [Ge84] presents an overview of the construction of the Gödel formula, and Chapter VII presents a detailed set of formulas that are used to construct the Gödel formula. Unfortunately, these formulas do not allow one to actually construct the Gödel formula, or even prove it exists. Most of Chapter VII, and some material in earlier chapters, has to be deeply rewritten to construct the Gödel formula. This chapter undertakes this task.

In Chapter V, Gensler describes the Gödel formula's father and the Gödel formula, which he denotes F and G. At this point, Gensler says that G is a result of a process performed on F, which he calls SON, but he does not give precise rules for SON, since he saves a deep level of precision for Chapter VII.

In Chapter VII, we see a series of precise definitions and expect a definition of SON as a function, the process promised in Chapter V, which would enable us to define G as $SON(F)$. However, the definition for $SON()$, and all the other formula definitions in Chapter VII, are as relations. The connection between F and G can only be expressed as $SON(F, G)$. This relational form specifies a condition that G would have to satisfy if it existed, but this is not enough to prove that G exists, much less construct G from F.

This revised version of Gensler's definitions develops $SON()$ as a function and defines G as $SON(F)$. This requires the revision of most of the definitions in Gensler's Chapter VII and some material in earlier chapters. It adds two new constructs to the language, using the same symbol set, and three new inference rules.

General conventions

Gensler's original definitions are in **black**.
The author's revised definitions are in **color**.
*Following each definition is a verbal paraphrase in **italics**.*
=df means "is defined as".

Syntactic categories

language	=df	a given set of symbols and a syntax for combining them; a formal language
system	=df	a given set of axioms and inference rules together with a language in which they are expressed; a formal system
string	=df	a finite sequence of symbols of the language
term	=df	a string representing a number
formula	=df	a string representing a statement; a well formed formula or wff
sequence	=df	a series of formulas joined by the formula separator (,)
proof	=df	a sequence in which each formula is either an axiom or an application of an inference rule to one or two previous formulas
theorem	=df	the last formula of a proof

Variable name conventions

The variables in the definitions below all refer to numeric ID's. For example, we say that a variable represents a formula when it acutally refers to the numeric ID of a formula, rather than the sequence of symbols in the formula.

$a, b, \ldots, g$	denote dependent quantities
h, j	denote (ID numbers of) numerals
k, m	denote (ID numbers of) sequences
p, q, r	denote (ID numbers of) formulas
s	denotes (an ID number of) a string
u	denotes (an ID number of) a variable name
x, y	denote (ID numbers of) terms

Symbols

Symbol	*ID*	*Meaning*
/	1	One; see below
+	2	Addition; conditional, loop
$*$	3	Multiplication (between terms); logical conjunction (between formulas); $**$ exponentiation; conditional
(	4	Left grouping; quantifier; loop
)	5	Right grouping; quantifier; loop
=	6	Equals; conditional; loop
¬	7	Logical negation; conditional
n	8	Variable names; see below
,	9	Formula separator; loop

Numerals are denoted with repeated /, e.g. 5 is denoted /////.

Variable names are formed with repeated n, e.g. n, nn, nnn, $nnnn$.

Quantifier: (u) means $(\forall u)$, i.e. that the following formula is true for all u.

Conditional: $* = f + = x \neg = y$ means that if p is true, then the value of the expression is x, otherwise it is y.

Loop: $(= u = x =)\ f \mathrel{++} y$ means a looped evaluation of a value. Initially store the value x to u. If p is true, then exit the loop. Otherwise evaluate y, and update u to the value of y. Continue evaluating p and updating u until p is true. The value of the whole expression is the final value of u.

In the definitions below, ID numbers of symbols are denoted with **underlining**: $\underline{/}$ $\underline{+}$ $\underline{*}$ $\underline{(}$ $\underline{)}$ $\underline{=}$ $\underline{\neg}$ $\underline{n}$, e.g. $\underline{n}$ means 8.

Alternate symbols

The following symbols are used in lieu of the original symbols for clarity.

1	=df /
…	
9	=df /////////
10	=df //////////
…	
$x(u)$	=df term x involving variable u
$p(u)$	=df formula p involving free variable u (Gensler's notation is $\ldots u \ldots$)
[ ]	=df ( )
{ }	=df ( )
$\uparrow$	=df $**$
$p \wedge q$	=df $p * q$
	$\wedge$ used only for conjunction, not multiplication
$p \vee q$	=df $\neg(\neg p \wedge \neg q)$
$p \rightarrow q$	=df $\neg p \vee q$
$(\exists x)\ p(x)$	=df $\neg(x)\ \neg p(x)$
IF	=df $* =$
THEN	=df $+ =$
ELSE	=df $\neg =$
FOR	=df (=
UNTIL	=df =)
DO	=df ++

Operator precedence from high to low

( )
$\uparrow$
$*$
$+$
$=$
$\neg$
$\wedge$
$\vee$
$\rightarrow$
(x)
$(\exists x)$
IF THEN ELSE
FOR UNTIL DO

Grammar rules

1. A string consisting of one or more / is a ***numeral*** and a ***term***.
2. A string consisting of one or more n is a ***variable*** and a ***term***.
3. If x and y are terms, then the string $(x + y)$ is a ***term***.
4. If x and y are terms, then the string $(x * y)$ is a ***term***.
5. If x and y are terms, then the string $(x \uparrow y)$ is a ***term***.
6. If x and y are terms, then the string $x = y$ is a ***formula***.
7. If f is a formula, then the string $\neg f$ is a ***formula***.
8. If f and g are a formulas, then the string $f \wedge g$ is a ***formula***.
9. If v is a variable and f is a formula, then the string $(v)\ f$ is a ***formula***.
10. If f is a formula and x and y are terms, then the string IF f THEN x ELSE y is a ***term***.
11. If u is a variable, x and y are terms, and f is a formula, then the string FOR $u = x$ UNTIL f DO y is a ***term***.
12. If f is a formula, then the string f is a ***formula sequence***.
13. If s and t are formula sequences, then the string s, t is a ***formula sequence***.

Axioms and their names

A1	$h = h$
CP1	$p \rightarrow p \wedge p$
CP2	$p \wedge q \rightarrow p$
CP3	$(p \rightarrow q) \rightarrow [\neg(q \wedge r) \rightarrow \neg(p \wedge r)]$
CQ1	$[(u)\ p(u)] \rightarrow [p(x)]$
CQ2	$(u)\ (p \rightarrow q) \rightarrow [(u)\ p \rightarrow (u)\ q]$
CQ3	$p \rightarrow (u)\ p$
C1	$x = y \rightarrow [(p(x)) \rightarrow (p(y))]$
CM1	$\neg(x + /) = /$
CM2	$(x + /) = (y + 1) \rightarrow x = y$
CM3	$\{p(/) \wedge (u)\ [p(u) \rightarrow p(u + /)]\} \rightarrow (u)\ p(u)$

Inference rules and their names

In the inference rules below:

$p \Leftrightarrow q$	means that p and q can be exchanged
$p \Rightarrow q$	means that p can be changed to q
$[p, q] \Rightarrow r$	means that p and q together can be changed to r

AR1	$(h + /) \Leftrightarrow h/$
AR2	$(h + /j) \Leftrightarrow (h/ + j)$
BR1	$(h * /) \Leftrightarrow h$
BR2	$(h * /j) \Leftrightarrow ((h * j) + h)$
BR3	$(h \uparrow j) \Leftrightarrow h$
BR4	$(h \uparrow /j) \Leftrightarrow ((h \uparrow j) * h)$
CCR1	$(\text{IF } p \text{ THEN } x \text{ ELSE } y, p) \Rightarrow x$
CCR2	$(\text{IF } p \text{ THEN } x \text{ ELSE } y, \neg p) \Rightarrow y$
CPR	$[(p \rightarrow q), p] \Rightarrow q$
CQR	$p \Rightarrow (u)\ p$
CLR	$(\text{FOR } u = x \text{ UNTIL } p(u) \text{ DO } y(u)) \Leftrightarrow$ $\text{IF } p(x) \text{ THEN } [\text{FOR } u = y(x) \text{ UNTIL } p(u) \text{ DO } y(u)] \text{ ELSE } x$

Function and relation definitions

$GR(x, y) \qquad =\text{df } (\exists a)\ x = y + a$

*Whether **x** is greater than **y***

$GE(x, y) \qquad =\text{df } GR(x + 1, y)$

*Whether **x** is greater than or equal to **y***

$N(s) \qquad =\text{df } (\exists a)\ 9 * x + 1 = 10 \uparrow a$

*Whether the string **s** is a numeral*

$NUM(s) \qquad =\text{df } \text{ FOR } \boldsymbol{n} = 1 \text{ UNTIL } GR(\boldsymbol{n}, 10 \uparrow s) \text{ DO } 10 * \boldsymbol{n} + 1$

*String **s** represented as a numeral*

$V(s) \qquad =\text{df } (\exists a)\ N(a) \wedge s = 8 * a$

*Whether the string **s** is a variable name*

$L(s) \qquad =\text{df } \text{ FOR } \boldsymbol{n} = 1 \text{ UNTIL } GE(s, 10 \uparrow \boldsymbol{n}) \wedge \neg GE(s, 10 \uparrow (\boldsymbol{n} + 1)) \text{ DO } \boldsymbol{n} + 1$

*Length of string **s***

$$
\begin{aligned}
s_1 s_2 \dots s_k \qquad =\text{df } & s_1 * (10 \uparrow (L(s_2) + \dots + L(s_k))) \\
& + s_2 * (10 \uparrow (L(s_3) + \dots + L(s_k))) \\
& + \dots \\
& + s_{k-1} * (10 \uparrow L(s_k)) \\
& + s_k
\end{aligned}
$$

Concatenation of strings $\boldsymbol{s_1}$ through $\boldsymbol{s_k}$

$$
\begin{aligned}
S(p, k, m) \qquad =\text{df } & (\exists a)\ m = kpa \\
& \wedge (k = \underline{,} \vee (\exists b)\ k = \underline{,}b\underline{,}) \\
& \wedge (a = \underline{,} \vee (\exists b)\ a = \underline{,}b\underline{,}) \\
& \wedge \neg(p = \underline{,}) \\
& \wedge \neg(\exists b)\ p = \underline{,}b
\end{aligned}
$$

$$\wedge \neg(\exists b)\ p = b\underline{,}$$

$$\wedge \neg(\exists b)(\exists c)\ p = b\underline{,}c$$

*Whether sequence **m** begins with sequence **k** immediately followed by formula **p***

$P(p,k,m)$ =df $(\exists a)(\exists b)\ m = kb \wedge GR(k,a) \wedge S(p,a,m)$

*Whether sequence **m** begins with sequence **k** and formula **p** follows later*

$WT(x,y,s)$ =df $(\exists a)\ [(\exists b)\ S(s,b,a) \wedge (b)(c)\{S(b,c,a) \rightarrow$

$$\{b = x\underline{=}y$$

$$\vee (\exists d)\ [P(d,c,a) \wedge b = \underline{\neg}d]$$

$$\vee (\exists d)(\exists e)\ [P(d,c,a) \wedge P(e,c,a) \wedge b = \underline{(}d\underline{\wedge}e\underline{)}]$$

$$\vee (\exists d)(\exists e)\ [V(d) \wedge P(e,c,a) \wedge b = \underline{(}d\underline{)}e]\}\}]$$

*Whether string **s** is a formula (wff) involving terms **x** and **y***

$T(s)$ =df $(\exists a)\ [(\exists b)\ S(s,b,a) \wedge (b)(c)\{S(b,c,a) \rightarrow N(b) \vee V(b) \vee$

$$(\exists d)(\exists e)\ [P(d,c,a) \wedge P(e,c,a) \wedge$$

$$\{b = \underline{(}d\underline{+}e\underline{)}$$

$$\vee\, b = \underline{(}d\underline{*}e\underline{)}$$

$$\vee\, b = \underline{(}d\underline{\uparrow}e\underline{)}$$

$$\vee\, [(\exists f)\ WT(d,e,f) \wedge$$

$$\{b = \underline{(}f\underline{,=}d\underline{,=}e\underline{)}] \vee (\exists g)\ [V(g) \wedge b = \underline{(}g\underline{,=}d\underline{,}f\underline{,=}e\underline{)}]\}]\}]\}]$$

*Whether string **s** is a term*

$W(s)$ =df $(\exists a)(\exists b)\ [T(a) \wedge T(b) \wedge WT(a,b,s)]$

*Whether string **s** is a formula (wff)*

$O(u,p,s)$ =df $V(u) \wedge W(p)$

$$\wedge \{p = s \vee (\exists a)\ [p = sa \wedge \neg a = \underline{n} \wedge \neg(\exists b)\ a = \underline{n}b]\}$$

$$\wedge \{s = u \vee (\exists a)\ [s = au \wedge \neg a = \underline{n} \wedge \neg(\exists b)\ a = b\underline{n}]\}$$

*Whether variable name **u** occurs at the end of string **s**, which begins formula **p***

$F(u,p,s)$ =df $O(u,p,s) \wedge \neg(\exists a)\ p = \underline{(}u\underline{)}a \wedge$

$$\begin{aligned}&(a)(b)(c)\ [W(a) \wedge b = c\underline{(}u\underline{)}a \wedge \{p = b \vee (\exists d)\ p = bd\}] \rightarrow\\&[GR(c,s) \vee GR(s,b)]\end{aligned}$$

Whether variable name ***u*** *is free in formula* ***p*** *and occurs at the end of string* ***s****, which begins* ***p***

$$\begin{aligned}SUB1T(x,y,p) \quad &=\text{df}\ \ \text{IF } [W(p) \wedge T(x) \wedge T(y)] \text{ THEN } [\text{IF } (\exists b)\ (p = yb) \text{ THEN } xb \text{ ELSE}\\&\qquad \{\text{IF } (\exists a)\ (p = ay) \text{ THEN } ax \text{ ELSE}\\&\qquad [\text{IF } (\exists a)(\exists b)\ (p = ayb) \text{ THEN } axb \text{ ELSE } p]\}] \text{ ELSE } p\\SUBT(x,y,p) \quad &=\text{df}\ \ \text{FOR } n = p \text{ UNTIL } \neg[W(p) \wedge T(x) \wedge T(y) \wedge\\&\qquad (\exists a)(\exists b)\ n = yb \vee n = ay \vee n = ayb] \text{ DO } SUB1T(x,y,n))\end{aligned}$$

The formula that results from substituting term ***x*** *for term* ***y*** *in formula* ***p***

$$\begin{aligned}U(s,x,u,p) \quad &=\text{df}\ (\exists c)\ [c = su \wedge F(u,p,c) \wedge\\&\qquad (d)(e)(f)\ \{[V(d) \wedge f = sx \wedge GR(e,s) \wedge \neg GR(e,f)] \rightarrow\\&\qquad F(d,p,e)\}]\\SUB1VF(x,u,p) \quad &=\text{df}\ \ \text{IF } [W(p) \wedge T(x) \wedge V(u)] \text{ THEN } [\text{IF } (\exists b)\ (p = ub) \text{ THEN } xb \text{ ELSE}\\&\qquad \{\text{IF } (\exists a)\ (p = au) \wedge U(a,x,u,p)] \text{ THEN } ax \text{ ELSE}\\&\qquad [\text{IF } (\exists a)(\exists b)\ (p = aub) \wedge U(a,x,u,p) \text{ THEN } axb \text{ ELSE } p]\}]\\&\qquad \text{ELSE } p\\SUBALLVF(x,u,p) \quad &=\text{df}\ \ \text{FOR } n = p \text{ UNTIL } \neg[W(p) \wedge T(x) \wedge V(u) \wedge \{(\exists a)(\exists b)\\&\qquad n = ub \vee [\{n = au \vee n = aub\} \wedge U(a,x,u,n)]\}]\\&\qquad \text{DO } SUB1VF(x,u,n))\end{aligned}$$

The formula that results from substituting term ***x*** *for free variable* ***u*** *in formula* ***p***

$$\begin{aligned}&A(p) \qquad =\text{df}\ W(p) \wedge\\&\text{[A1]} \qquad \{(\exists a)\ [T(a) \wedge p = a\underline{=}a]\\&\qquad\qquad \vee (\exists a)(\exists b)(\exists c)\ [W(a) \wedge W(b) \wedge W(c) \wedge\\&\text{[CP1]} \qquad \{p = \underline{\neg(}a\underline{\wedge\neg(}a\underline{\wedge}a\underline{))}\\&\text{[CP2]} \qquad \vee\, p = \underline{\neg((}a\underline{\wedge}b\underline{)\wedge\neg}a\underline{)}\\&\text{[CP3]} \qquad \vee\, p = \underline{\neg(\neg(}a\underline{\wedge\neg}b\underline{)\wedge\neg\neg(\neg(}b\underline{\wedge}c\underline{)\wedge\neg\neg(}c\underline{\wedge}a\underline{)))}\}]\\&\text{[CQ1]} \qquad \vee (\exists a)(\exists b)(\exists c)(\exists d)\ [a = SUBALLVF(c,d,b) \wedge p = \underline{\neg((}d\underline{)}b\underline{\wedge\neg}a\underline{)}]\\&\text{[CQ2]} \qquad \vee (\exists a)(\exists b)(\exists c)\ [V(a) \wedge W(b) \wedge W(c) \wedge\\&\qquad\qquad\qquad p = \underline{\neg((}a\underline{)\neg(}b\underline{\wedge\neg}c\underline{)\wedge\neg\neg((}a\underline{)}b\underline{\wedge\neg(}a\underline{)}c\underline{))}]\end{aligned}$$

[CQ3] $\lor (\exists a)(\exists b)\ [V(a) \land W(b) \land \neg(\exists c)\ F(a,b,c) \land p = \underline{\neg(}b\underline{\land\neg(}a\underline{)}b\underline{)}]$

[C1] $\lor (\exists a)(\exists b)(\exists c)(\exists d)(\exists e)(\exists f)\ [a = SUBALLVF(b,e,f) \land$
$d = SUBALLVF(c,e,f) \land p = \underline{\neg(}b\underline{=}c\underline{\land\neg\neg(}a\underline{\land\neg}d\underline{))}]$

[CM1] $\lor (\exists a)\ [T(a) \land p = \underline{\neg(}a\underline{+/)=/}]$

[CM2] $\lor (\exists a)(\exists b)\ [T(a) \land T(b) \land p = \underline{\neg((}a\underline{+/)=(}b\underline{+/)\land\neg}a\underline{=}b\underline{)}]$

[CM3] $\lor (\exists a)(\exists b)(\exists c)(\exists d)(\exists e)\ [b = SUBALLVF(\underline{/},a,c) \land e = \underline{(}a\underline{+/)} \land$
$d = SUBALLVF(e,a,c) \land p = \underline{\neg((}b\underline{\land(}a\underline{)\neg(}c\underline{\land\neg}d\underline{))\land\neg(}a\underline{)}c\underline{)}]\}$

*Whether formula **p** is an axiom*

$R(p,q,r)$ =df $(\exists a)(\exists b)(\exists c)(\exists d)\ [p = SUBT(a,b,q) \land$

[AR1] $\{[a = \underline{(}c\underline{+/)} \land b = c\underline{/}]$

[AR2] $\lor [a = \underline{(}c\underline{+/}d\underline{)} \land b = \underline{(}c\underline{/+}d\underline{)}]$

[BR1] $\lor [a = \underline{(}c\underline{*/)} \land b = c]$

[BR2] $\lor [a = \underline{(}c\underline{*/}d\underline{)} \land b = \underline{((}c\underline{*}d\underline{)+}c\underline{)}]$

[BR3] $\lor [a = \underline{(}c\underline{\uparrow/)} \land b = c]$

[BR4] $\lor [a = \underline{(}c\underline{\uparrow/}d\underline{)} \land b = \underline{((}c\underline{\uparrow}d\underline{)*}c]$

[CCR1] $\lor [a = \underline{(}\ \underline{\text{IF}}\ r\ \underline{\text{THEN}}\ c\ \underline{\text{ELSE}}\ d\underline{)} \land b = c]$

[CCR2] $\lor [a = \underline{(}\ \underline{\text{IF}}\ \underline{\neg}r\ \underline{\text{THEN}}\ c\ \underline{\text{ELSE}}\ d\underline{)} \land b = d]\}]$

[CPR] $\lor [p = \underline{\neg(}q\underline{\land(\neg}r\underline{)}]$

[CQR] $\lor (\exists a)\ [V(a) \land p = \underline{(}a\underline{)}q]$

[CLR] $\lor (\exists a)(\exists b)(\exists c)(\exists d)\ [V(a) \land T(b) \land W(c) \land T(d) \land$
$q = \underline{(}a\underline{,=}b\underline{,}c\underline{,=}d\underline{)} \land$
$\underline{(}p = \underline{(}\ \underline{\text{IF}}\ SUBALLVF(b,a,c)\underline{\text{THEN}}$
$\underline{(}\ \underline{\text{FOR}}\ a\underline{=}SUBT(b,a,d)\ \underline{\text{UNTIL}}\ c\ \text{DO}\ d\underline{)}\ \underline{\text{ELSE}}\ b\underline{))}]$

*Whether formula **p** is the result of applying an inference rule to formulas **q** and **r***

$TH(p)$ =df $(\exists a)\ [(\exists b)\ S(p,b,a) \land (b)(c)\ \{S(b,c,a) \rightarrow$
$A(b) \lor (\exists d)(\exists e)\ [P(d,c,a) \land P(e,c,a) \land R(b,d,e)]\}]$

*Whether formula **p** is a theorem*

$SON(p)$ =df $SUBALLVF(NUM(p), \underline{n}, p)$

*Son of formula **p**, the result of substituting **n** with the number for **p** in the formula **p***

F =df $\neg TH(SON(\underline{n}))$

*Father of the Gödel formula, which states that the son of formula number **n** is not provable*

G =df $SON(F)$

The Gödel formula, which asserts its own unprovability and thereby proves Gödel's incompleteness theorem

SYSTEM D

System D starts with System C and adds the Gödel formula G from System C to System D as an axiom. This gives us a modified definition of $A()$, which implicitly affects the definitions of $TH()$ and F and gives us a new G.

In general, it is obvious that adding symbols, axioms, and inference rules to System C, while it affects the final form of the definitions for System C, does not affect the fact that they exist. For any axiomatic system which includes System C, there is always some Gödel formula G which asserts its own unprovability, and the resulting system is always incomplete.

Gödel's Theorem thus applies not just to System C but to a wide variety of axiomatic systems, including widely used and practical systems such as those that handle negative numbers, fractions, real numbers, complex numbers, algebra, calculus, differential equations, matrices, etc.

COMPLETENESS, CONSISTENCY, AND REDUCIBILITY

Reduction vs. completeness

Gensler paraphrases Gödel's Theorem as *Arithmetic cannot be reduced to any axiomatic system*, but Gensler's concept of reduction is somewhat different from Gödel's concept of completeness. Gensler briefly explains this late in his book (Chapter VI, p. 64). Here we examine this difference in greater detail and also examine the related concepts of consistency and soundness.

System properties

We use the following definitions.

- ***Statement*** means a formula in the language of the system.
- A statement is ***provable*** if there is a proof of the statement that uses only the axioms and inference rules of the system.
- A system is ***complete*** if, for every statement, either it or its negation is provable.
- A system is ***reducible*** if every true statement is provable and no false statement is provable.
- A system is ***consistent*** if it is never possible to prove both a statement and its negation.
- A system is ***sound*** if it every provable statement is true.
- A system is ***reflective*** if every true statement is provable. (Unfortunately, the usual term for such a system is ***complete***. We introduce the term ***reflective*** to avoid the obvious collision with the above sense of ***complete***.)

Below are symbolic definitions of the above five system properties. These definitions use the same conventions and definitions as above:

- p denotes any formula in the language of the system
- $TH(p)$ means p is provable in the system (also denoted $S \vdash p$, where S is the system)
- $\neg$ means negation
- $\wedge$ means conjunction
- $\vee$ means disjunction.

We use these symbols even if the language of the system does not contain them, since we determine these properties from outside the system.

complete =df $(p)\ TH(p) \vee TH(\neg p)$
*A system is **complete** if every statement or its negation is provable.*

consistent =df $(p)\ \neg(TH(p) \wedge TH(\neg p))$
*A system is **consistent** if a statement and its negation are never both provable.*

sound =df $(p)\ TH(p) \rightarrow p$
*A system is **sound** if every provable statement is true.*

reflective =df $(p)\ p \rightarrow TH(p)$
*A system is **reflective** if every true statement of the system is provable.*

reducible =df $(p)\ TH(p) \leftrightarrow p$
*A system is **reducible** if every provable statement is true and every true statement is provable.*

Relation of properties

Completeness and consistency are purely formal properties. They refer only to the provability of the statements of a system, which as we have seen above depends only on the axioms and inference rules of the system.

Soundness, reflectivity, and reducibility are not purely formal. They refer to the truth or falsehood of the statements of the system, which, as we have also seen above, depends on the interpretation of the language in addition to the formal properties of the system. However, the systems we are examining have a canonical, standard interpretation. Unless otherwise specified, when we refer to a system having or not having one of these properties, we assume the standard interpretation.

A determination of completeness and consistency can be made only if we have a negation operator in the system. The language of Systems A, B, E, and F do not include negation, so we cannot determine whether they are complete or consistent.

A number of dependencies can be located within these five properties.

For all systems:

- sound $\wedge$ reflective $\leftrightarrow$ reducible
- unsound $\vee$ unreflective $\leftrightarrow$ irreducible

For systems whose language includes negation:

- sound $\rightarrow$ consistent
- inconsistent $\rightarrow$ unsound
- reflective $\rightarrow$ complete
- incomplete $\rightarrow$ unreflective
- reducible $\rightarrow$ consistent $\wedge$ complete
- inconsistent $\vee$ incomplete $\rightarrow$ irreducible
- sound $\wedge$ complete $\rightarrow$ reducible
- irreducible $\rightarrow$ unsound $\vee$ incomplete
- reducible $\rightarrow$ consistent $\wedge$ sound
- inconsistent $\vee$ unsound $\rightarrow$ irreducible

The Gödel formula and System C

It is widely believed, but has not been proved, that System C is consistent and sound. If this is the case, then the Gödel formula is true, and System C is incomplete.

If the Gödel formula is false, then System C contains an inconsistency. If this were actually found, it would be big news. It would mean that the standard system that mathematicians use to reason about positive whole numbers contains a serious flaw that would render it suspect and all the proofs based on it questionable. However, even if this were to happen, Gödel's Theorem would not fail!

Hardly anyone believes that System C is actually inconsistent. For this reason, many people claim that Gödel's Theorem shows that arithmetic is incomplete, when it actually shows that arithmetic is either incomplete or inconsistent.

Gödel's Theorem does show that System C is irreducible. Here we show the two paths for the Gödel formula G that both lead to System C irreducibility.

- G is true (widely believed) $\rightarrow$ G is unprovable $\rightarrow$ System C is incomplete $\rightarrow$ System C is unreflective $\rightarrow$ System C is irreducible
- G is false (widely disbelieved) $\rightarrow$ G is provable $\rightarrow$ System C is inconsistent $\rightarrow$ System C is unsound $\rightarrow$ System C is irreducible

Systems A_2, B_2, E_2, and F_2

Systems A_2, B_2, E_2, and F_2 are modified forms of Systems A, B, E, and F respectively. For each modified system, we make two additions: the symbol $\neg$ for logical negation, and an axiom named AM which states that $\neg(x + y) = y$ for any terms x and y.

These two changes enable us to state and prove inequalities of specific numbers. Both the modified and original system allow us to prove equalities of specific numbers. Given any two numbers, the modified systems allow us to prove that they are either equal or unequal.

Systems E_3 and F_3

Systems E_3 and F_3 are modified forms of Systems E_2 and F_2. We form these systems by adding an axiom EX which reads $\neg / = /$. Axiom EX is false in the standard interpretation, and since its negation $/ = /$ is also provable in both these systems, each system contains an inconsistency.

System attributes under the standard interpretation

System	**Complete**	**Consistent**	**Sound**	**Reflective**	**Reducible**
A	—	—	Yes	Yes	Yes
A_2	Yes	Yes	Yes	Yes	Yes
B	—	—	Yes	Yes	Yes
B_2	Yes	Yes	Yes	Yes	Yes
C	No	Yes?	Yes?	No?	No

D	No?	Yes?	Yes?	No?	No
E	—	—	Yes	No	No
E_2	No	Yes	Yes	No	No
E_3	No	No	No	No	No
F	—	—	Yes	Yes	Yes
F_2	Yes	Yes	Yes	Yes	Yes
F_3	Yes	No	No	Yes	No

? = widely believed to be such but not proved

— = not applicable since language does not include negation

System attributes under alternate interpretations

Systems	**Interpretation**	**Sound**	**Reflective**	**Reducible**
A, B, F	1	Yes	Yes	Yes
	2	Yes	Yes	Yes
	3	No	No	No
	4	No	Yes	No
A_2, B_2, F_2	1	Yes	Yes	Yes
	2	Yes	Yes	Yes
	3	No	No	No
	4	No	No	No
C	1	Yes?	No?	No?
	2	No	No	No
	3	No	No	No
	4	No	No	No
D	1	Yes?	No?	No?
	2	No	No	No
	3	No	No	No
	4	No	No	No
E	1	Yes	No	No
	2	Yes	Yes	Yes
	3	No	Yes	No
	4	No	Yes	No
E_2	1	Yes	No	No
	2	Yes	No	No
	3	No	No	No
	4	No	No	No
E_3	1	No	No	No
	2	No	No	No
	3	No	No	No

	4	No	No	No
F_3	1	No	Yes	No
	2	No	No	No
	3	No	No	No
	4	No	No	No

? = widely believed to be such but not proved

THE LIAR PARADOX

The proof of Gödel's Theorem constructs a self-referential statement, the Gödel formula, in the language of System C. The Gödel formula asserts its own unprovability and can be paraphrased *This statement is not provable*. Closely related to the Gödel formula is the Liar Paradox, the statement *This statement is false*.

In a little known proof, Gödel showed that, unlike the Gödel formula, the Liar Paradox cannot be constructed in the language of System C. The proof is very short, once we have all the definitions that we used to construct the Gödel formula, and we present it now.

The proof is by contradiction. Assume that there is a formula $TR()$ in the language of System C such that, for any formula ID p, $TR(p)$ is true if and only if the formula that p represents is true. This would mean that truth, which we have already seen to be dependent on interpretation and determined outside of System C, can somehow be determined by a formula written in the language of System C.

Then we use the $SON()$ function defined above to construct a father-son pair K and L, the father of the Liar Paradox and the Liar Paradox itself.

K =df $\neg TR(SON(\underline{n}))$

L =df $SON(K)$

K says that the son of formula n is false. To obtain L, we substitute the n in K with the ID for K. Then L states that the son of K is false. But the son of K is L, so L asserts that L is false. If L is true, then it is false, and if it is false, then it is true. Since L cannot be either true or false, there is no such formula $TR()$ in the language of System C, and there is no way to actually construct L in the language of System C.

This shows that the Liar Paradox cannot be constructed in the language of System C or any similar formal mathematical language. Gödel further argues that the Liar Paradox cannot be constructed in *any* language. This is because, as we saw before when we considered interpretation, a statement of the truth of a statement cannot be made in the language of the original statement.

This proof of the nonexistence of the Liar's Paradox in a formal system appears in section 7 of [G34], a set of notes from a lecture Gödel gave in 1934 which were not published until 1965. Alfred Tarski independently discovered and published the same theorem with a different proof in his 1936 paper [T36], from which the theorem is generally known as Tarski's Undefinability Theorem.

REFERENCES

D M. Davis, *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems, and Computable Functions*, Raven, 1965, http://books.google.com/....

Ge84 H. J. Gensler, *Gödel's Theorem Simplified*, University Press of America, 1984, http://books.google.com/....

Ge10 H. J. Gensler, *Introduction to Logic*, Routledge, 2010, http://books.google.com/....

G31a K. Gödel, Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I, *Monatshefte für Mathematik und Physik*, **38** (1931) 173–198.

G31b K. Gödel, tr. J. van Heijenoort, On formally undecidable propositions of Principia Mathematica and related systems I; translation of [G31a] approved by Gödel with emendations and supplemental note, published in [G86] and [H].

G34 K. Gödel, On undecidable propositions of formal mathemtical systems; 1934 lecture, notes taken by S. C. Kleene and J. B. Rosser, approved by Gödel with emendations, published in [D] and [G86].

G86 K. Gödel, ed. S. Feferman, *Collected Papers, Volume 1: Publications 1929– 1936*, Oxford, 1986, http://books.google.com/....

H J. van Heijenoort, ed., *From Frege to Gödel: A Source Book in Mathematical Logic*, Harvard, 1967, http://books.google.com/....

T36 A. Tarski, Der Wahrheitsbegriff in den formalisierten Sprachen, *Studia Philosophica*, **1** (1936) 261–405.

T83 A. Tarski, tr. J. H. Woodger, The concept of truth in formalized languages; translation of [T36], in A. Tarski, ed. J. Corcoran, *Logic, Semantics, Metamathematics: Papers from 1923 to 1938*, Hackett, 1983, http://books.google.com/....