Steeple #2:
Gödel's First *In*completeness Theorem
(The "Parlor Trick" Theorem)

(and thereafter: Steeple #3: The "Silver Blaze" (from Sherlock Holmes) Theorem)

Selmer Bringsjord

Are Humans Rational?

Dec 5 2019

RPI

Troy NY USA

Background Context …

Gödel's Great Theorems (OUP)

by Selmer Bringsjord

- Introduction ("The Wager")
- Brief Preliminaries (e.g. the propositional calculus & FOL)
- The Completeness Theorem
- The First Incompleteness Theorem
- The Second Incompleteness Theorem
- The Speedup Theorem
- The Continuum-Hypothesis Theorem
- The Time-Travel Theorem
- Gödel's "God Theorem"
- Could a Machine Match Gödel's Genius?

Gödel's Great Theorems (OUP)

by Selmer Bringsjord

- Introduction ("The Wager")
- Brief Preliminaries (e.g. the propositional calculus & FOL)
- The Completeness Theorem
- The First Incompleteness Theorem
- The Second Incompleteness Theorem
- The Speedup Theorem
- The Continuum-Hypothesis Theorem
- The Time-Travel Theorem
- Gödel's "God Theorem"
- Could a Machine Match Gödel's Genius?

Gödel's Great Theorems (OUP)

by Selmer Bringsjord

- Introduction ("The Wager")
- Brief Preliminaries (e.g. the propositional calculus & FOL)
- The Completeness Theorem
- The First Incompleteness Theorem
- The Second Incompleteness Theorem
- The Speedup Theorem
- The Continuum-Hypothesis Theorem
- The Time-Travel Theorem
- Gödel's "God Theorem"
- Could a Machine Match Gödel's Genius?

A corollary of the First Incompleteness Theorem: We cannot prove that mathematics is consistent.

Gödel's Great Theorems (OUP)

by Selmer Bringsjord

- Introduction ("The Wager")
- Brief Preliminaries (e.g. the propositional calculus & FOL)
- The Completeness Theorem
- The First Incompleteness Theorem
- The Second Incompleteness Theorem
- The Speedup Theorem
- The Continuum-Hypothesis Theorem
- The Time-Travel Theorem
- Gödel's "God Theorem"
- Could a Machine Match Gödel's Genius?

Gödel's Great Theorems (OUP)

by Selmer Bringsjord

- Introduction ("The Wager")
- Brief Preliminaries (e.g. the propositional calculus & FOL)
- The Completeness Theorem
- The First Incompleteness Theorem
- The Second Incompleteness Theorem
- The Speedup Theorem
- The Continuum-Hypothesis Theorem
- The Time-Travel Theorem
- Gödel's "God Theorem"
- Could a Machine Match Gödel's Genius?

Based on the Sherlock Holmes mystery "Silver Blaze"; read for next (& last) class).

Some Timeline Points

1978 Princeton NJ USA.

1940 Back to USA, for good.

1936 Schlick murdered; Austria annexed

1933 Hitler comes to power.

1930 Announces (First) *In*completeness Theorem

1929 Doctoral Dissertation: Proof of Completeness Theorem

Undergrad in seminar by Schlick

1923 Vienna

1906 Brünn, Austria-Hungary

Some Timeline Points

1978 Princeton NJ USA.

1940 Back to USA, for good.

1936 Schlick murdered; Austria annexed

1933 Hitler comes to power.

1930 Announces (First) *In*completeness Theorem

1929 Doctoral Dissertation: Proof of Completeness Theorem

Undergrad in seminar by Schlick

1923 Vienna

1906 Brünn, Austria-Hungary

Some Timeline Points

1978 Princeton NJ USA.

1940 Back to USA, for good.

1936 Schlick murdered; Austria annexed

1933 Hitler comes to power.

1930 Announces (First) *In*completeness Theorem

1929 Doctoral Dissertation: Proof of Completeness Theorem

Undergrad in seminar by Schlick

1923 Vienna

1906 Brünn, Austria-Hungary

Some Timeline Points

1978 Princeton NJ USA.

1940 Back to USA, for good.

1936 Schlick murdered; Austria annexed

1933 Hitler comes to power.

1930 Announces (First) *In*completeness Theorem

1929 Doctoral Dissertation: Proof of Completeness Theorem

Undergrad in seminar by Schlick

1923 Vienna

1906 Brünn, Austria-Hungary

Steeple #2 (*In*completeness) …

Goal: Put *you* in position to prove Gödel's first incompleteness theorem!

Goal: Put *you* in position to prove Gödel's first incompleteness theorem!

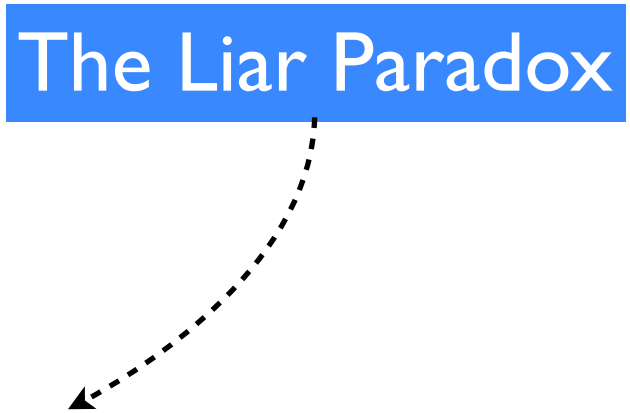
We have the background (if).

The “Liar Tree”

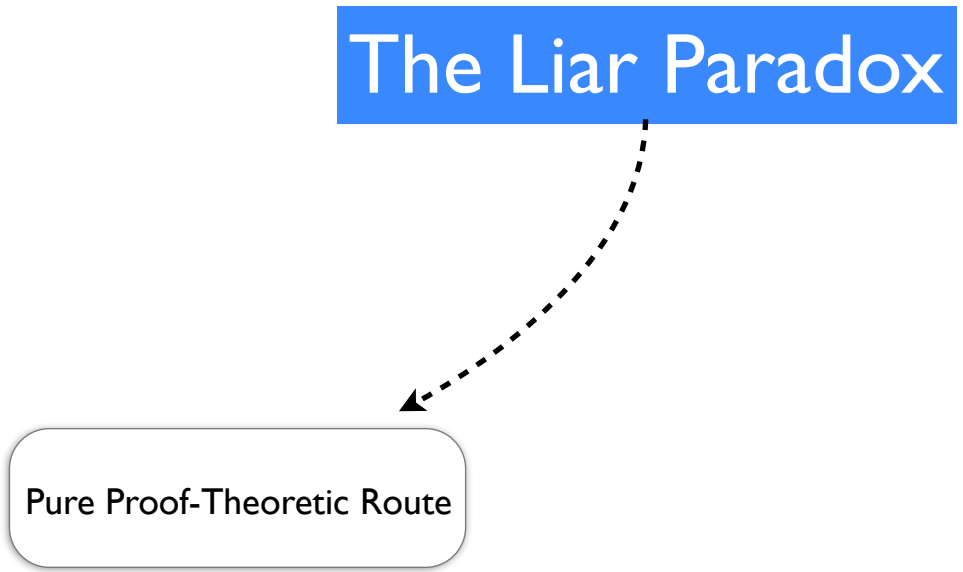
The “Liar Tree”

The Liar Paradox

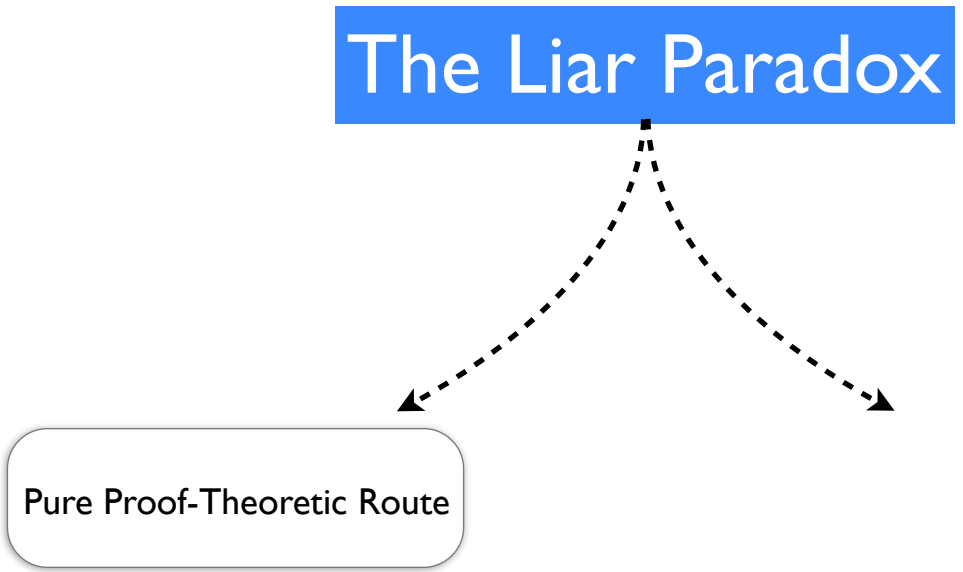
The “Liar Tree”



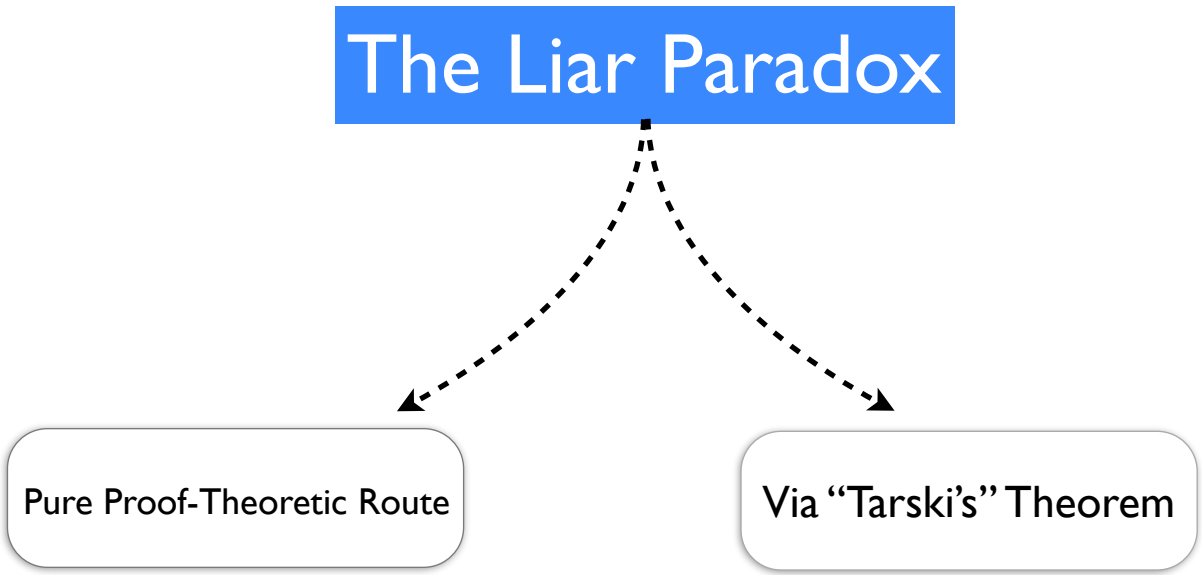
The “Liar Tree”



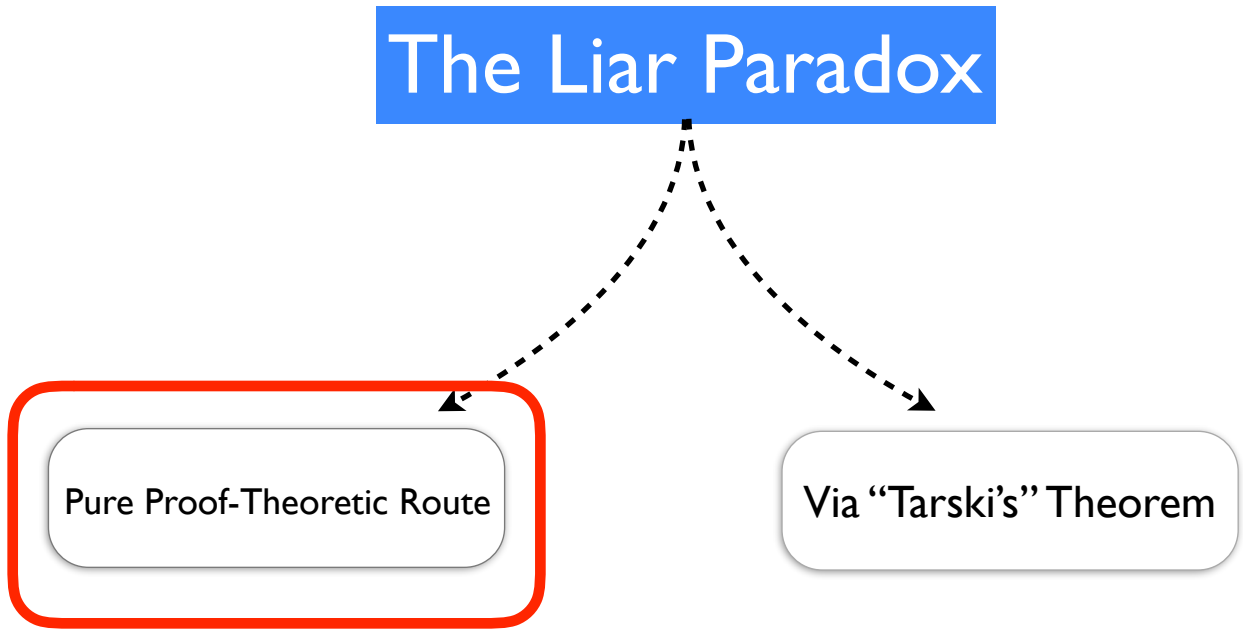
The “Liar Tree”



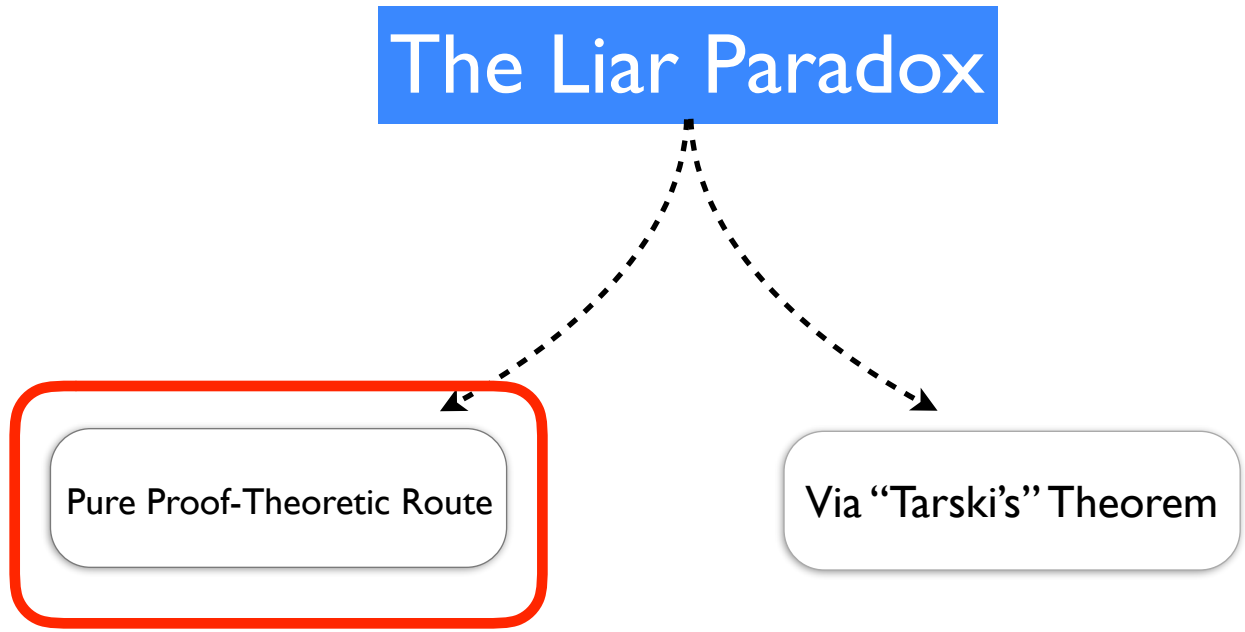
The "Liar Tree"



The "Liar Tree"



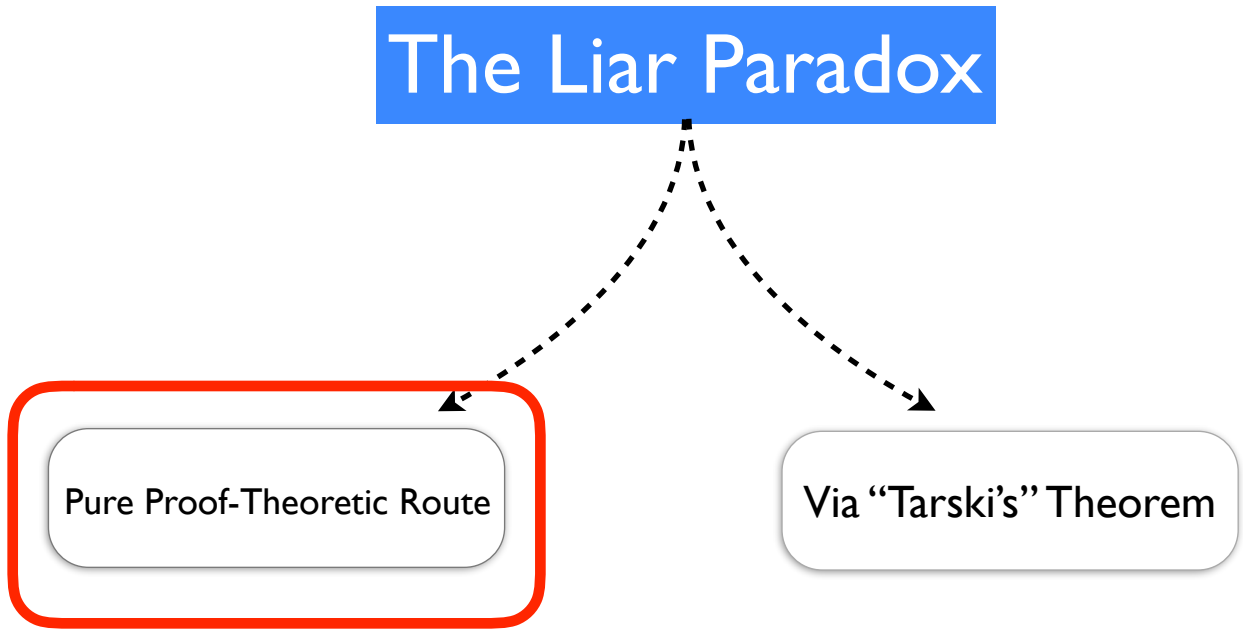
The "Liar Tree"



Paul Erdős

"The Book"

The "Liar Tree"



Paul Erdős

"The Book"

Ergo, step one: What *is* LP?

Remember …
“The (Economical) Liar” …

Remember …
"The (Economical) Liar" …

L: This sentence is false.

Remember …
"The (Economical) Liar" …

L: This sentence is false.

Suppose that T(**L**); then ¬T(**L**).

Remember …
"The (Economical) Liar" …

L: This sentence is false.

Suppose that T(**L**); then ¬T(**L**).

Suppose that ¬T(**L**) then T(**L**).

Remember …
"The (Economical) Liar" …

L: This sentence is false.

Suppose that T(**L**); then ¬T(**L**).

Suppose that ¬T(**L**) then T(**L**).

Hence: T(**L**) iff (i.e., if & only if) ¬T(L).

Remember …
"The (Economical) Liar" …

L: This sentence is false.

Suppose that T(**L**); then ¬T(**L**).

Suppose that ¬T(**L**) then T(**L**).

Hence: T(**L**) iff (i.e., if & only if) ¬T(L).

Contradiction!

The “Gödelian” Liar

The "Gödelian" Liar

$\bar{P}$: This sentence is unprovable.

The "Gödelian" Liar

$\bar{P}$: This sentence is unprovable.

Suppose that $\bar{P}$ is true. Then we can immediately deduce that $\bar{P}$ is provable, because here is a proof: $\bar{P} \rightarrow \bar{P}$ is an easy theorem, and from it and our supposition we deduce $\bar{P}$ by *modus ponens*. But since what $\bar{P}$ says is that it's unprovable, we have deduced that $\bar{P}$ is false under our initial supposition.

The "Gödelian" Liar

$\bar{P}$: This sentence is unprovable.

Suppose that $\bar{P}$ is true. Then we can immediately deduce that $\bar{P}$ is provable, because here is a proof: $\bar{P} \rightarrow \bar{P}$ is an easy theorem, and from it and our supposition we deduce $\bar{P}$ by *modus ponens*. But since what $\bar{P}$ says is that it's unprovable, we have deduced that $\bar{P}$ is false under our initial supposition.

Suppose on the other hand that $\bar{P}$ is false. Then we can immediately deduce that $\bar{P}$ is unprovable: Suppose for *reductio* that $\bar{P}$ *is* provable; then $\bar{P}$ holds as a result of some proof, but what $\bar{P}$ says is that it's unprovable; and so we have contradiction. But since what $\bar{P}$ says is that it's unprovable, and we have just proved that under our supposition, we arrive at the conclusion that $\bar{P}$ is true.

The "Gödelian" Liar

$\bar{P}$: This sentence is unprovable.

Suppose that $\bar{P}$ is true. Then we can immediately deduce that $\bar{P}$ is provable, because here is a proof: $\bar{P} \rightarrow \bar{P}$ is an easy theorem, and from it and our supposition we deduce $\bar{P}$ by *modus ponens*. But since what $\bar{P}$ says is that it's unprovable, we have deduced that $\bar{P}$ is false under our initial supposition.

Suppose on the other hand that $\bar{P}$ is false. Then we can immediately deduce that $\bar{P}$ is unprovable: Suppose for *reductio* that $\bar{P}$ *is* provable; then $\bar{P}$ holds as a result of some proof, but what $\bar{P}$ says is that it's unprovable; and so we have contradiction. But since what $\bar{P}$ says is that it's unprovable, and we have just proved that under our supposition, we arrive at the conclusion that $\bar{P}$ is true.

T($\bar{P}$) iff (i.e., if & only if) ¬T($\bar{P}$) = F($\bar{P}$)

The "Gödelian" Liar

$\bar{P}$: This sentence is unprovable.

Suppose that $\bar{P}$ is true. Then we can immediately deduce that $\bar{P}$ is provable, because here is a proof: $\bar{P} \rightarrow \bar{P}$ is an easy theorem, and from it and our supposition we deduce $\bar{P}$ by *modus ponens*. But since what $\bar{P}$ says is that it's unprovable, we have deduced that $\bar{P}$ is false under our initial supposition.

Suppose on the other hand that $\bar{P}$ is false. Then we can immediately deduce that $\bar{P}$ is unprovable: Suppose for *reductio* that $\bar{P}$ *is* provable; then $\bar{P}$ holds as a result of some proof, but what $\bar{P}$ says is that it's unprovable; and so we have contradiction. But since what $\bar{P}$ says is that it's unprovable, and we have just proved that under our supposition, we arrive at the conclusion that $\bar{P}$ is true.

T($\bar{P}$) iff (i.e., if & only if) ¬T($\bar{P}$) = F($\bar{P}$)

Contradiction!

Next, recall:
PA (Peano Arithmetic):

$$
\begin{array}{ll}
\text{A1} & \forall x(0 \neq s(x)) \\
\text{A2} & \forall x \forall y(s(x) = s(y) \rightarrow x = y) \\
\text{A3} & \forall x(x \neq 0 \rightarrow \exists y(x = s(y)) \\
\text{A4} & \forall x(x + 0 = x) \\
\text{A5} & \forall x \forall y(x + s(y) = s(x + y)) \\
\text{A6} & \forall x(x \times 0 = 0) \\
\text{A7} & \forall x \forall y(x \times s(y) = (x \times y) + x)
\end{array}
$$

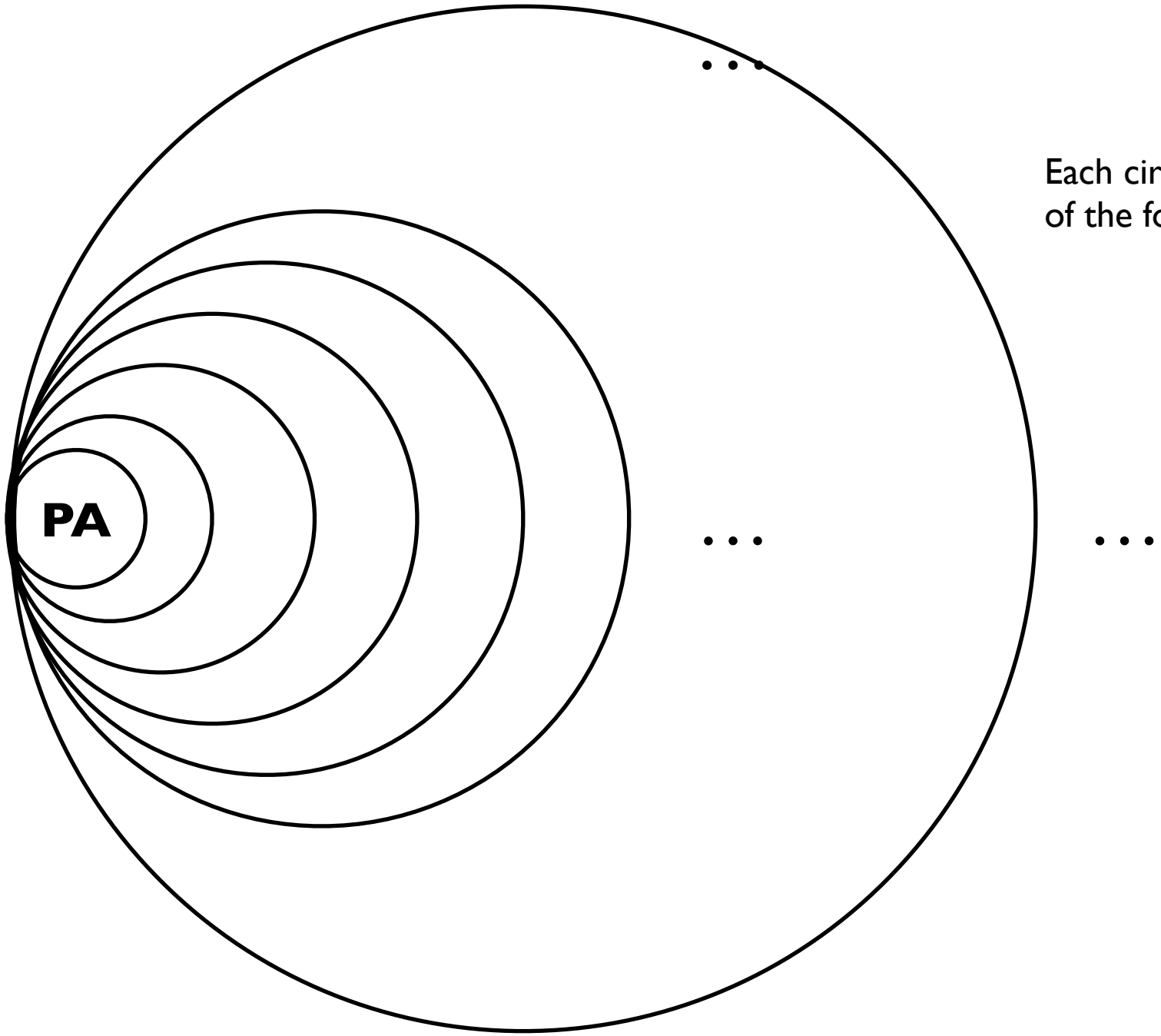
And, every sentence that is the universal closure of an instance of

$$([\phi(0) \wedge \forall x(\phi(x) \rightarrow \phi(s(x))] \rightarrow \forall x \phi(x))$$

where $\phi(x)$ is open wff with variable x, and perhaps others, free.

Arithmetic is Part of All Things Sci/Eng/Tech!

and courtesy of Gödel: We can't even prove all truths of arithmetic!



Each circle is a larger part of the formal sciences.

Gödel Numbering

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Gödel Numbering

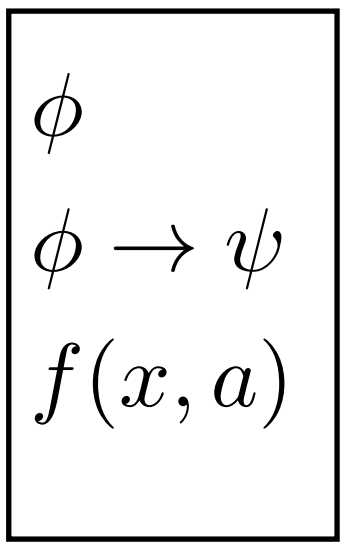
Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

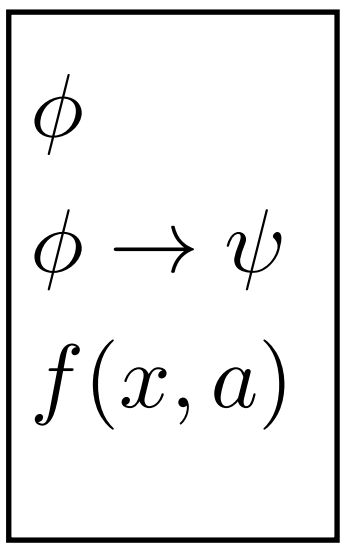
Solution: Gödel numbering



Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering

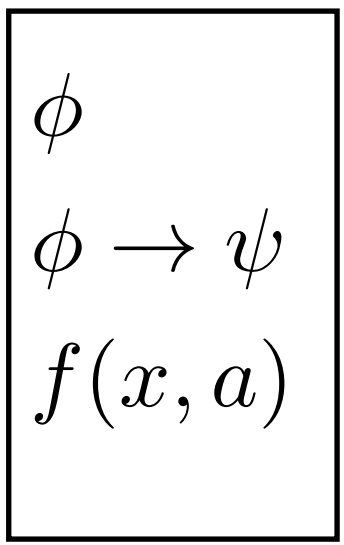


Syntactic objects

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering



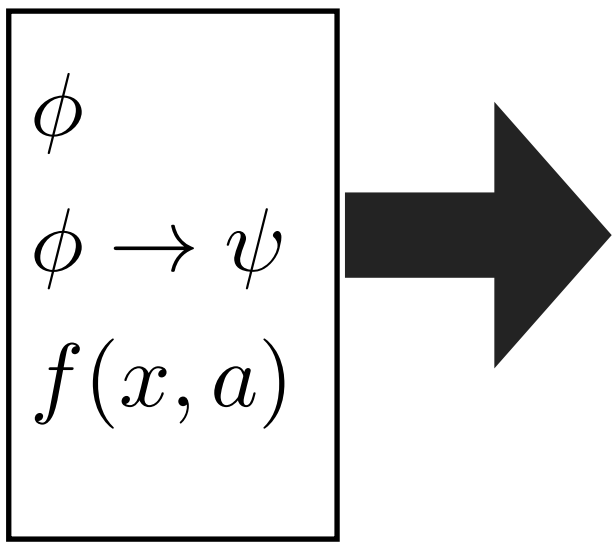
Syntactic objects

(formulae, terms, proofs etc)

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering



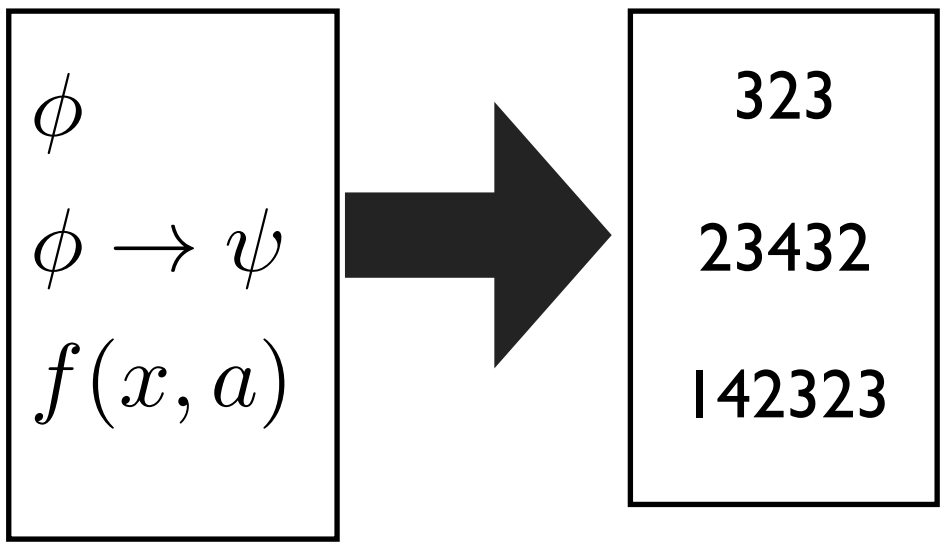
Syntactic objects

(formulae, terms, proofs etc)

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering



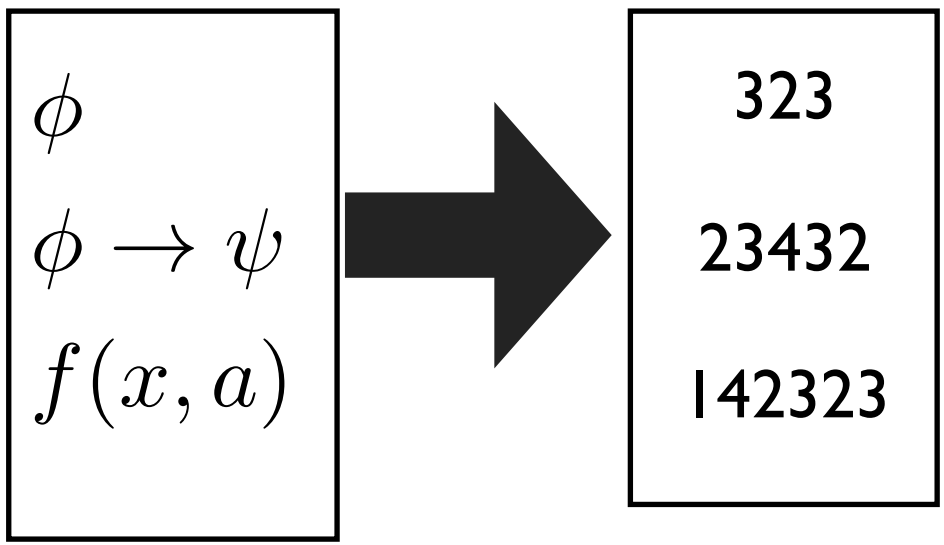
Syntactic objects

(formulae, terms, proofs etc)

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering



Syntactic objects

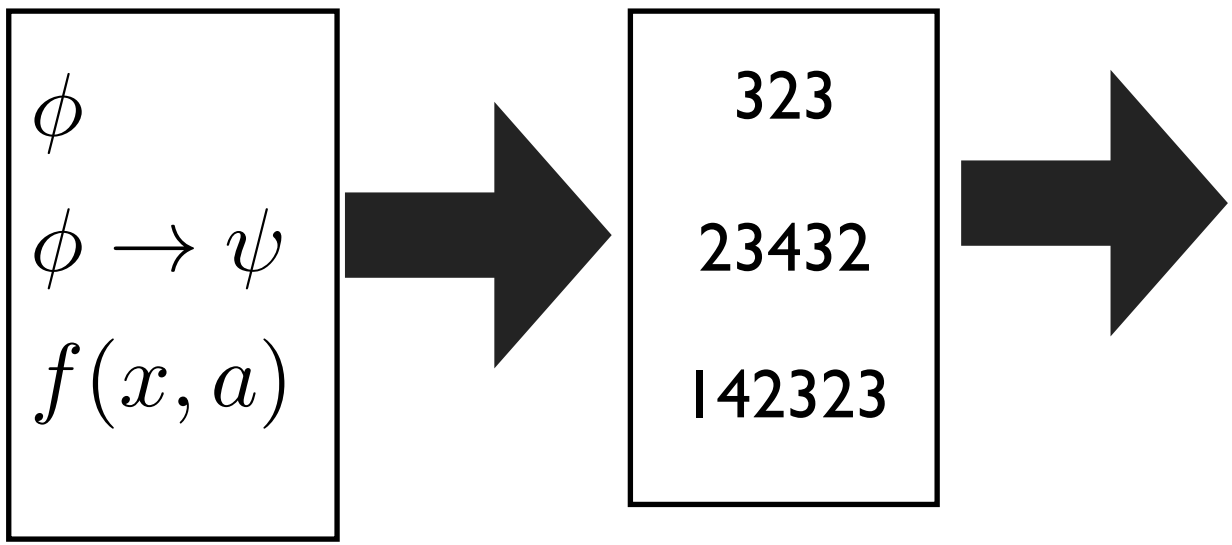
Gödel number

(formulae, terms, proofs etc)

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering



Syntactic objects

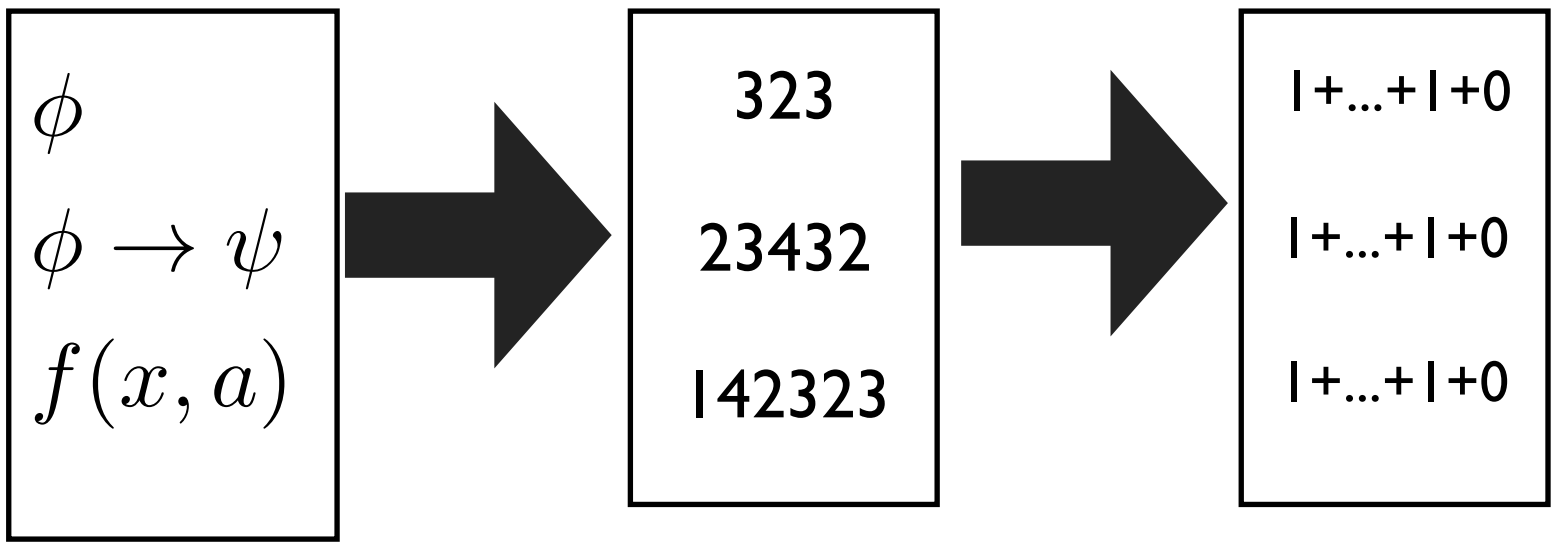
Gödel number

(formulae, terms, proofs etc)

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering



Syntactic objects

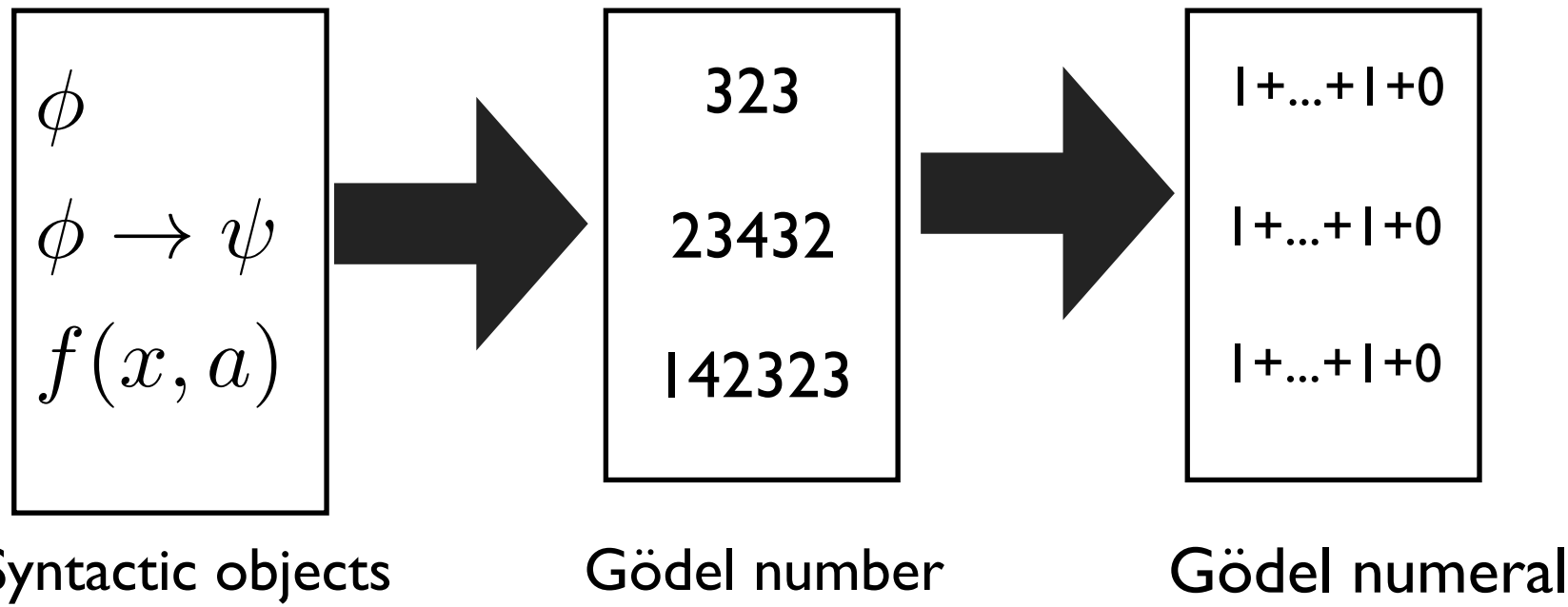
Gödel number

(formulae, terms, proofs etc)

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering

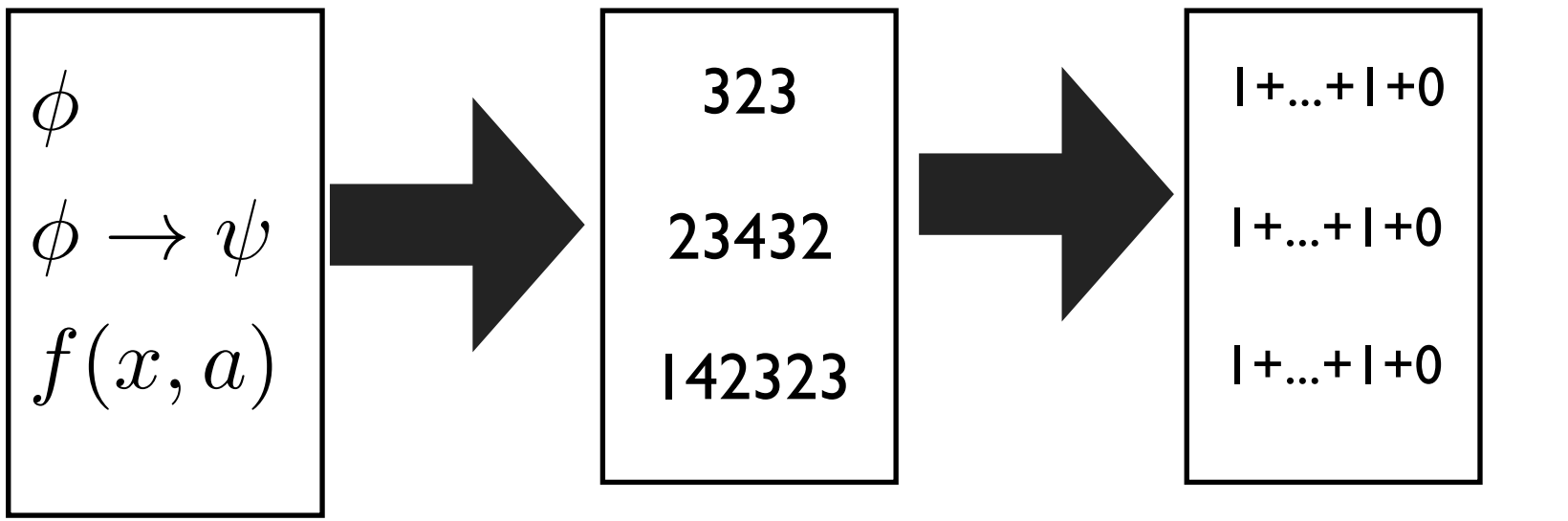


(formulae, terms, proofs etc)

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering



Syntactic objects

(formulae, terms, proofs etc)

Gödel number

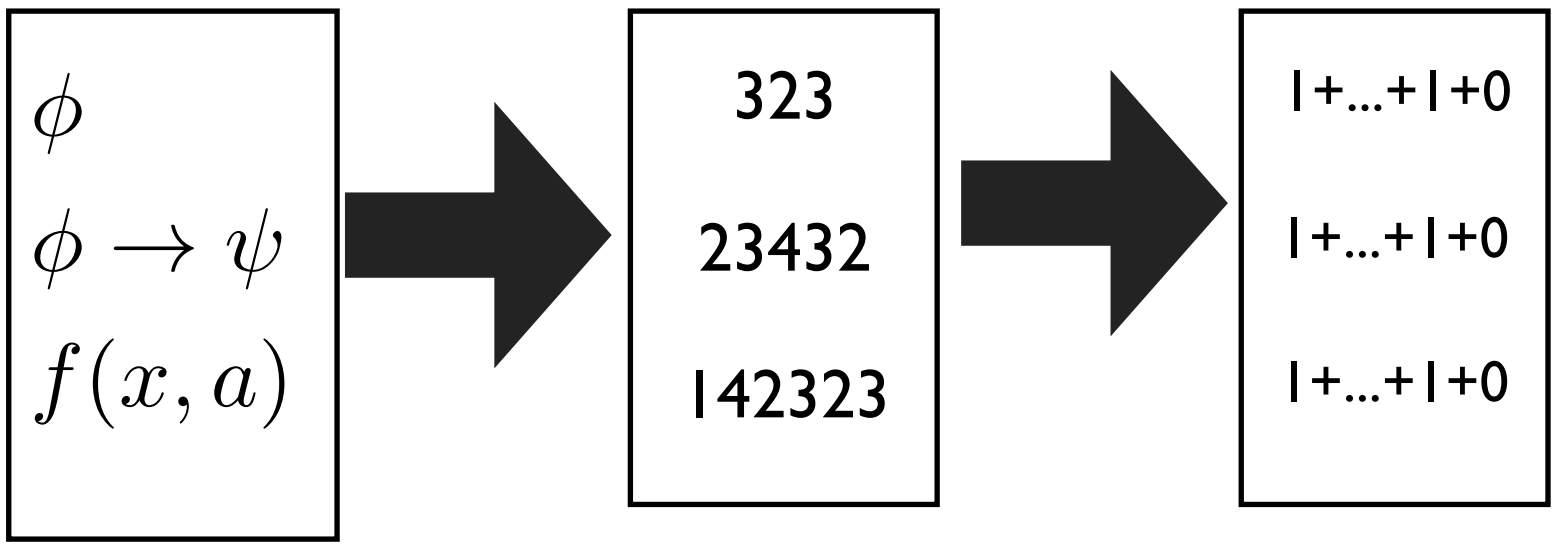
Gödel numeral

back to syntax

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering



Syntactic objects

(formulae, terms, proofs etc)

ϕ

Gödel number

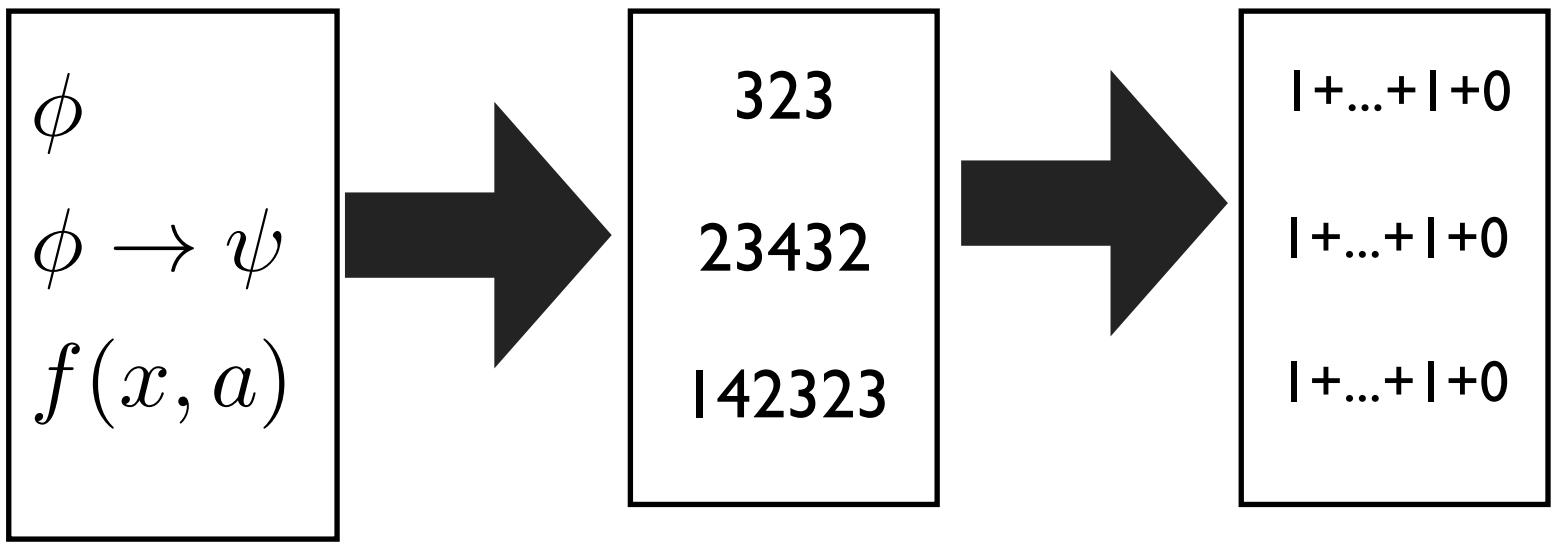
Gödel numeral

back to syntax

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering



Syntactic objects

Gödel number

Gödel numeral

(formulae, terms, proofs etc)

back to syntax

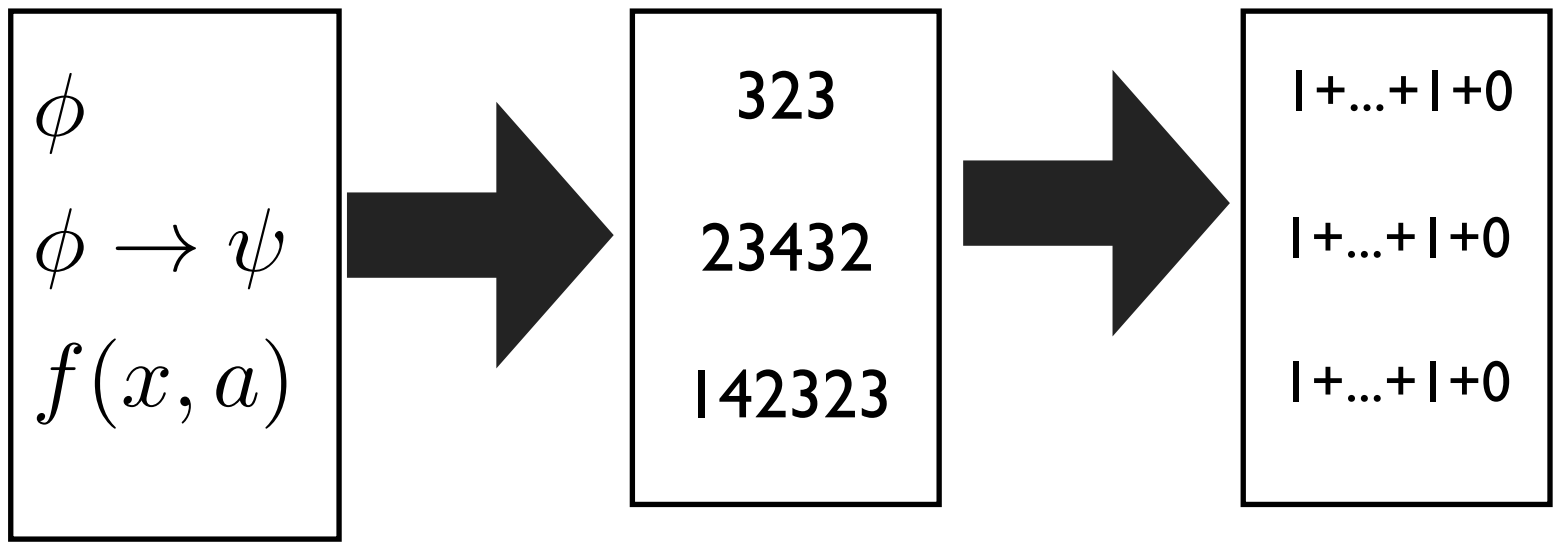
ϕ

n^{ϕ}

Gödel Numbering

Problem: How do enables a formula to refer to other formula and itself (and also other objects like proofs, terms etc.), in a perfectly consistent way?

Solution: Gödel numbering



Syntactic objects

(formulae, terms, proofs etc)

ϕ

Gödel number

n^{ϕ}

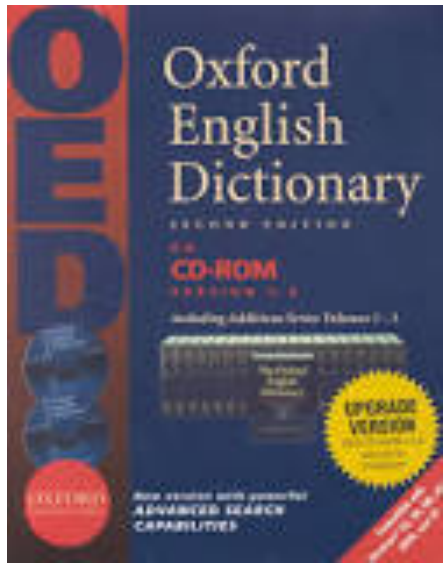
Gödel numeral

back to syntax

$\hat{n}^{\phi}$ (or just "ϕ")

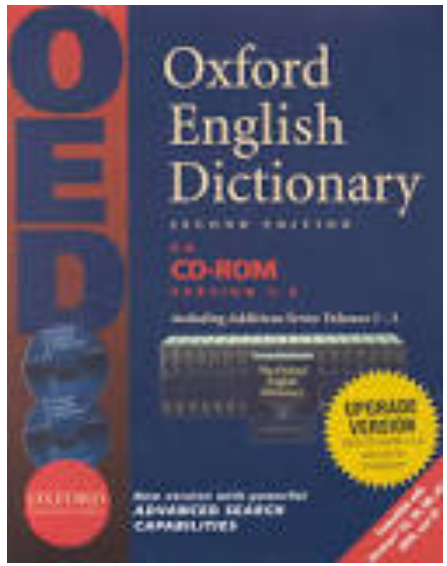
Gödel Numbering, the Easy Way

Just realize that every entry in a dictionary is named by a number *n*, and by the same basic lexicographic ordering, every computer program, formula, etc. is named by a number *m* in a lexicographic ordering going from 1, to 2, to …



Gödel Numbering, the Easy Way

Just realize that every entry in a dictionary is named by a number *n*, and by the same basic lexicographic ordering, every computer program, formula, etc. is named by a number *m* in a lexicographic ordering going from 1, to 2, to …



So, `gimcrack` is named by some positive integer *k*.
Hence, I can just refer to this word as "*k*".

Gödel Numbering, the Easy Way

Just realize that every entry in a dictionary is named by a number *n*, and by the same basic lexicographic ordering, every computer program, formula, etc. is named by a number *m* in a lexicographic ordering going from 1, to 2, to …

Gödel Numbering, the Easy Way

Just realize that every entry in a dictionary is named by a number *n*, and by the same basic lexicographic ordering, every computer program, formula, etc. is named by a number *m* in a lexicographic ordering going from 1, to 2, to …

Or, every syntactically valid computer program in Haskell that you will ever write can be uniquely picked about some number *m* in the lexicographic ordering of all syntactically valid such programs, and then we can just use a numeral or string "m" (or whatever) to refer to your program in a formal language/logic!

Gödel's First Incompleteness Theorem

Suppose that elementary arithmetic (i.e., **PA**) is *consistent* (no contradiction can be derived in it) and *program-decidable* (there's a program *P* that, given as input an arbitrary formula *p*, can decide whether or not *p* is in **PA**).

Then there is sentence *g** in the language of elementary arithmetic which is such that:

*g** can't be proved from **PA** (i.e., not **PA** |- *g**)**!**

And, not-*g** can't be proved from **PA** either (i.e., not **PA** |- not-*g**)!

Gödel's First Incompleteness Theorem

Suppose that elementary arithmetic (i.e., **PA**) is *consistent* (no contradiction can be derived in it) and *program-decidable* (there's a program *P* that, given as input an arbitrary formula *p*, can decide whether or not *p* is in **PA**).

Then there is sentence *g** in the language of elementary arithmetic which is such that:

*g** can't be proved from **PA** (i.e., not **PA** |- *g**)**!**

And, not-*g** can't be proved from **PA** either (i.e., not **PA** |- not-*g**)!

(Oh, and: *g** is true**!)**

Proof Kernel for Theorem GI

Part 1: Recipe *R*

Let *q*(*x*) be an arbitrary formula of arithmetic with one open variable *x*. (E.g., *x* + 3 = 5. And here *q*(2) would be 2 + 3 = 5.)

Gödel invented a recipe *R* that, given any *q*(*x*) as an ingredient template that you are free to choose, produces a self-referential formula *g* such that:

PA |- *g* <=> *q*("*g*")

(i.e., a formula *g* that says: "I have property *q*!")

Proof Kernel for Theorem GI

Part 2: Follow Recipe R, Guided by The Liar

First, for $q(x)$ we choose a formula q^* that holds of any "s" if and only if s can be proved from **PA**; i.e.,

PA |- q^*("s") iff **PA** |- s (1)

Next, we follow Gödel's Recipe *R* to build a g^* such that:

PA |- g^* <=> not-q^*("g^*") (2)

Proof Kernel for Theorem GI

Part 2: Follow Recipe R, Guided by The Liar

First, for $q(x)$ we choose a formula q^* that holds of any "s" if and only if s can be proved from **PA**; i.e.,

PA |- q^*("s") iff **PA** |- s (1)

Next, we follow Gödel's Recipe *R* to build a g^* such that:

PA |- g^* <=> not-q^*("g^*") (2)

g^* thus says: "I'm not true!"/"I'm not provable."
And so, the key question (assignment!): **PA** |- g^*?!?

Indirect Proof

GI: *g** isn't provable from **PA**; nor is the negation of *g**!

Proof: Let's follow The Liar: Suppose that *g** *is* provable from PA; i.e., suppose **PA** |- *g**. Then by (1), with g* substituted for s, we have:

PA |- *q**("*g**") iff **PA** |- *g** (1')

From our supposition and working right to left by *modus ponens* on (1') we deduce:

PA |- *q**("*g**") (3.1)

But from our supposition and the earlier (see previous slide) (2), we can deduce by *modus ponens* that from PA the opposite can be proved! I.e., we have:

PA |- not-*q**("*g**") (3.2)

But (3.1) and (3.2) together means that **PA** is inconsistent, since it generates a contradiction. But we are working under the supposition that **PA** is consistent. Hence by indirect proof *g** is *not* provable from **PA**.

Indirect Proof

GI: *g** isn't provable from **PA**, nor is the negation of *g**!

Proof: Let's follow The Liar: Suppose that *g** *is* provable from PA; i.e., suppose **PA** |- *g**. Then by (1), with g* substituted for s, we have:

PA |- *q**("*g**") iff **PA** |- *g** (1')

From our supposition and working right to left by *modus ponens* on (1') we deduce:

PA |- *q**("*g**") (3.1)

But from our supposition and the earlier (see previous slide) (2), we can deduce by *modus ponens* that from PA the opposite can be proved! I.e., we have:

PA |- not-*q**("*g**") (3.2)

But (3.1) and (3.2) together means that **PA** is inconsistent, since it generates a contradiction. But we are working under the supposition that **PA** is consistent. Hence by indirect proof *g** is *not* provable from **PA**.

What about the second option? Can you follow The Liar to show that supposing that the negation of *g** (i.e., not-*g**) is provable from **PA** also leads to a contradiction, and hence can't be?

“Silly abstract nonsense! There aren’t any concrete examples of g*!”

Ah, but: Goodstein's Theorem!

Ah, but: Goodstein's Theorem!

The Goodstein Sequence goes to zero!

Pure base *n* representation of a number *r*

- Represent *r* as only sum of powers of *n* in which the exponents are also powers of n etc

$$266 = 2^{2^{(2^{2^0}+2^0)}} + 2^{(2^{2^0}+2^0)} + 2^{2^0}$$

Grow Function

$Grow_k(n)$:

1. Take the pure base k representation of n
2. Replace all k by $k+1$. Compute the number obtained.
3. Subtract one from the number

Example of **Grow**

$$Grow_2(19)$$

$$19 = 2^{2^{2^{2^0}}} + 2^{2^0} + 2^0$$

$$3^{3^{3^{3^0}}} + 3^{3^0} + 3^0$$

$$3^{3^{3^{3^0}}} + 3^{3^0} + 3^0 - 1$$

$$7625597484990$$

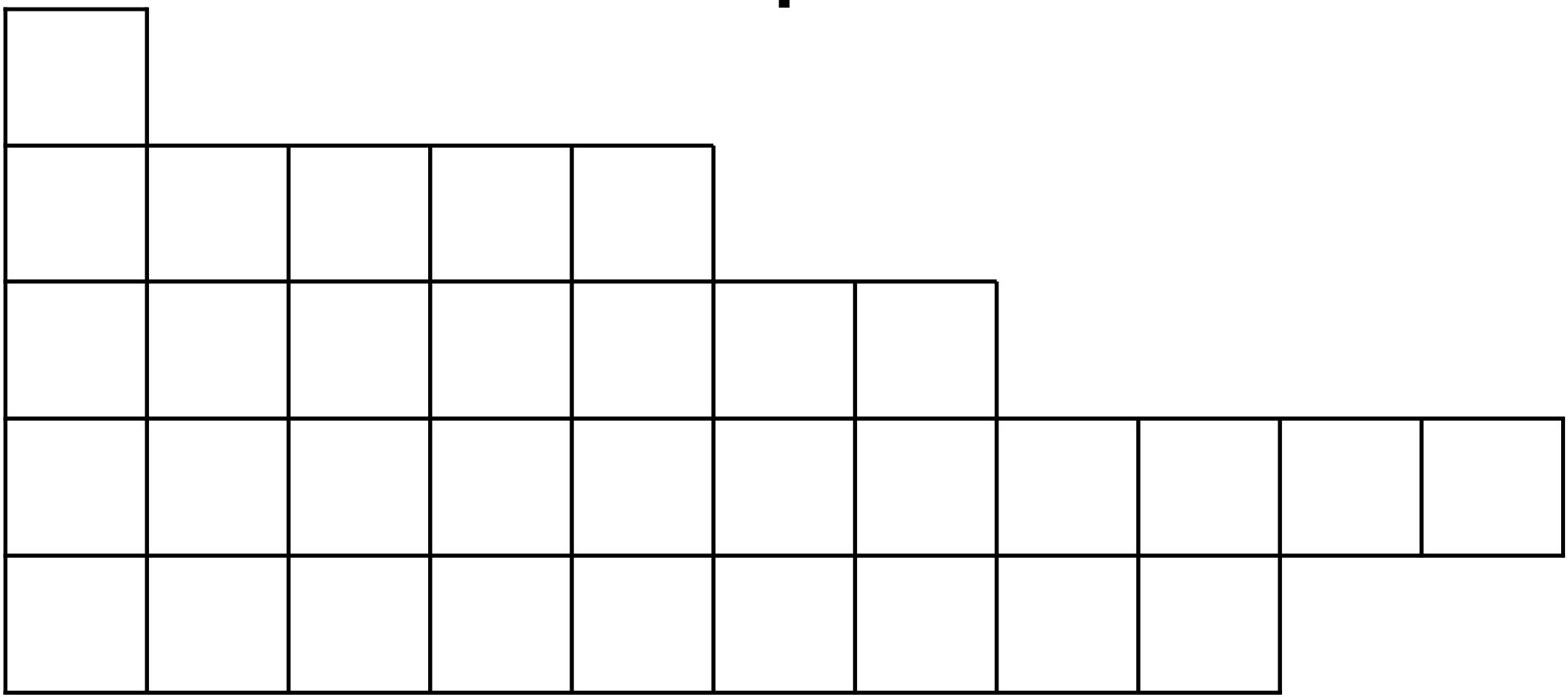
Goodstein Sequence

- For any natural number *m*

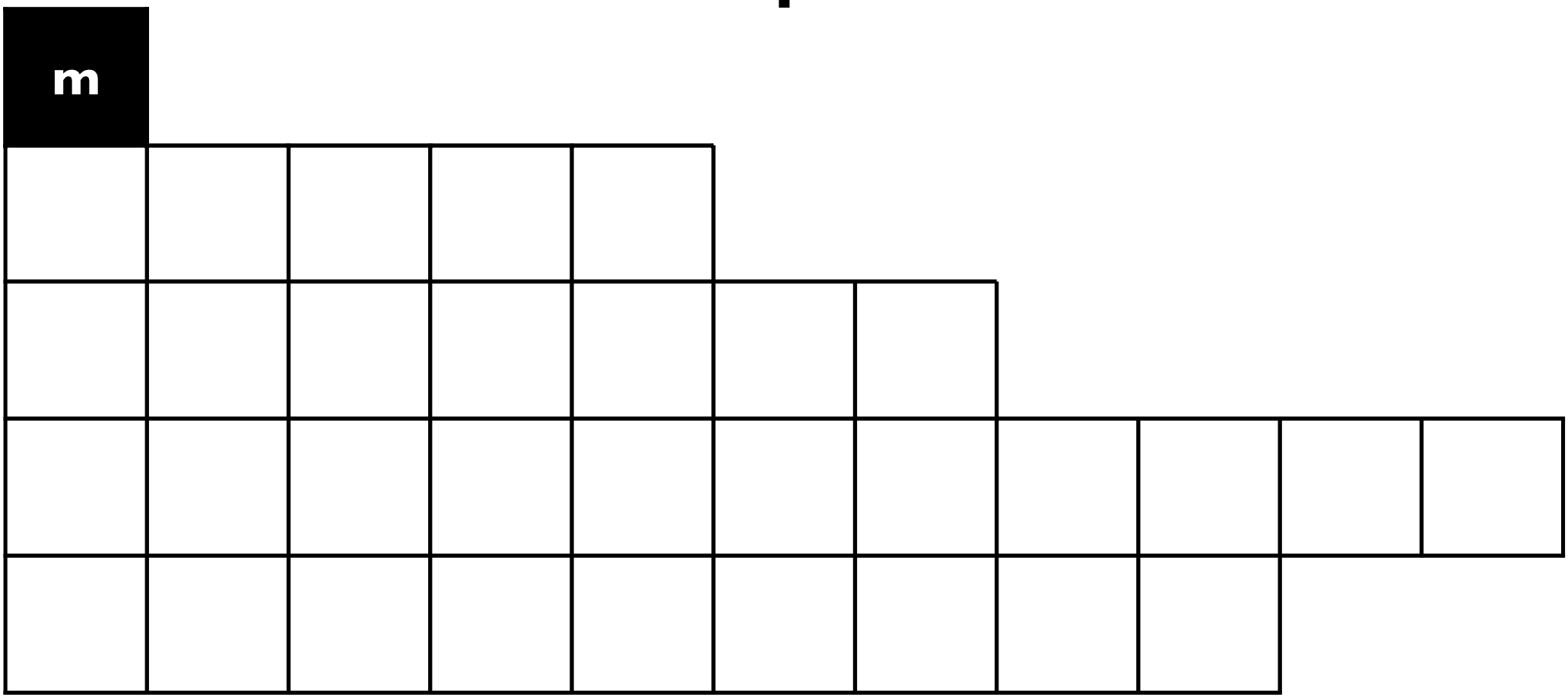
$$
\begin{aligned}
&m \\
&Grow_2(m) \\
&Grow_3(Grow_2(m)) \\
&Grow_4(Grow_3(Grow_2(m))), \\
&\ldots
\end{aligned}
$$

Sample Values

Sample Values



Sample Values



Sample Values

m										
2	2	2	1	0						

Sample Values

m										
2	2	2	1	0						
3	3	3	3	2	1	0				

Sample Values

m										
2	2	2	1	0						
3	3	3	3	2	1	0				
4	4	26	41	60	83	109	139	...	11327 (96th term)	...

Sample Values

m										
2	2	2	1	0						
3	3	3	3	2	1	0				
4	4	26	41	60	83	109	139	...	11327 (96th term)	...
5	15	$\sim 10^{13}$	$\sim 10^{155}$	$\sim 10^{2185}$	$\sim 10^{36306}$	10^{695975}	$10^{15151337}$	...		

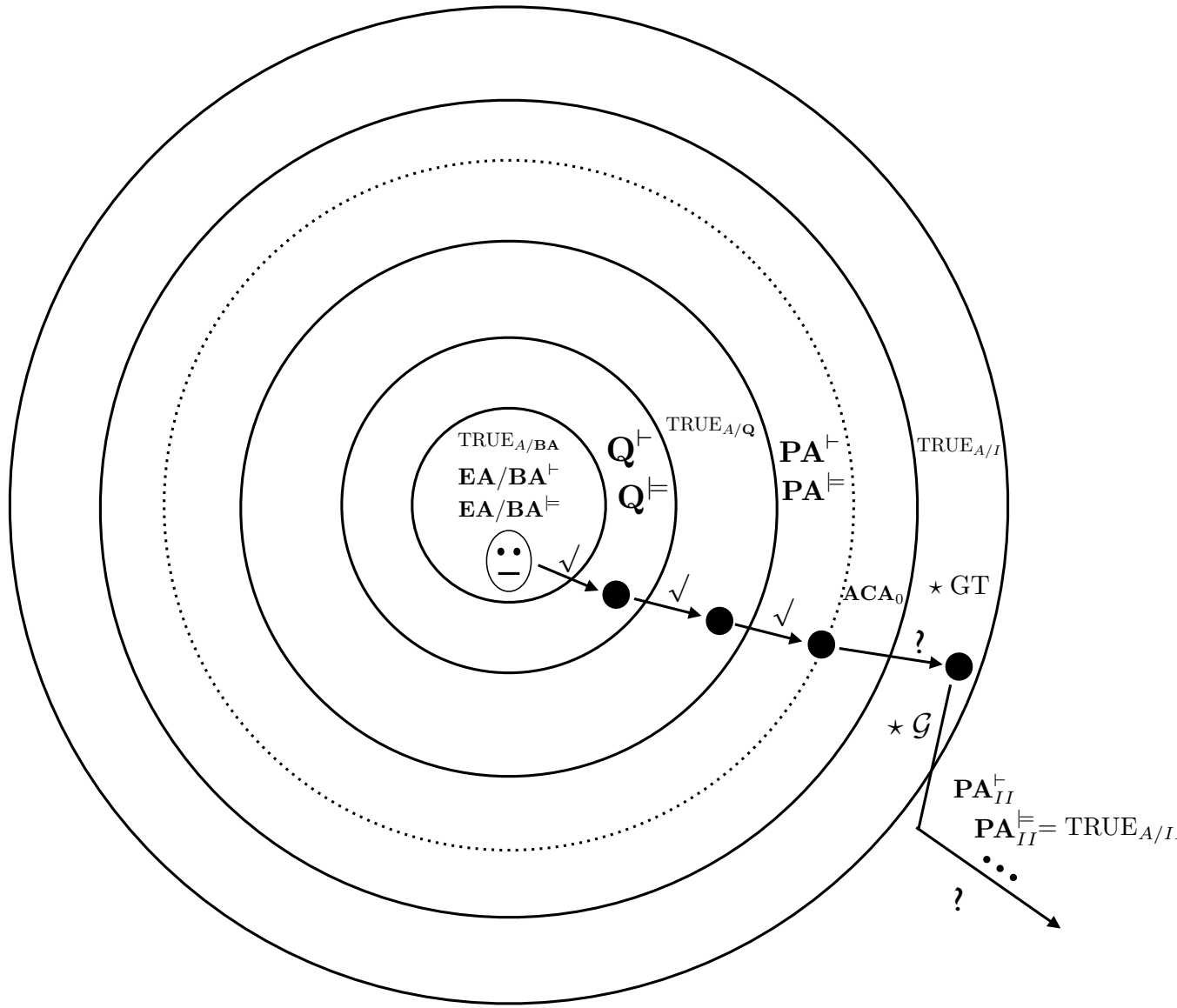
Ah, but: Goodstein's Theorem!

Ah, but: Goodstein's Theorem!

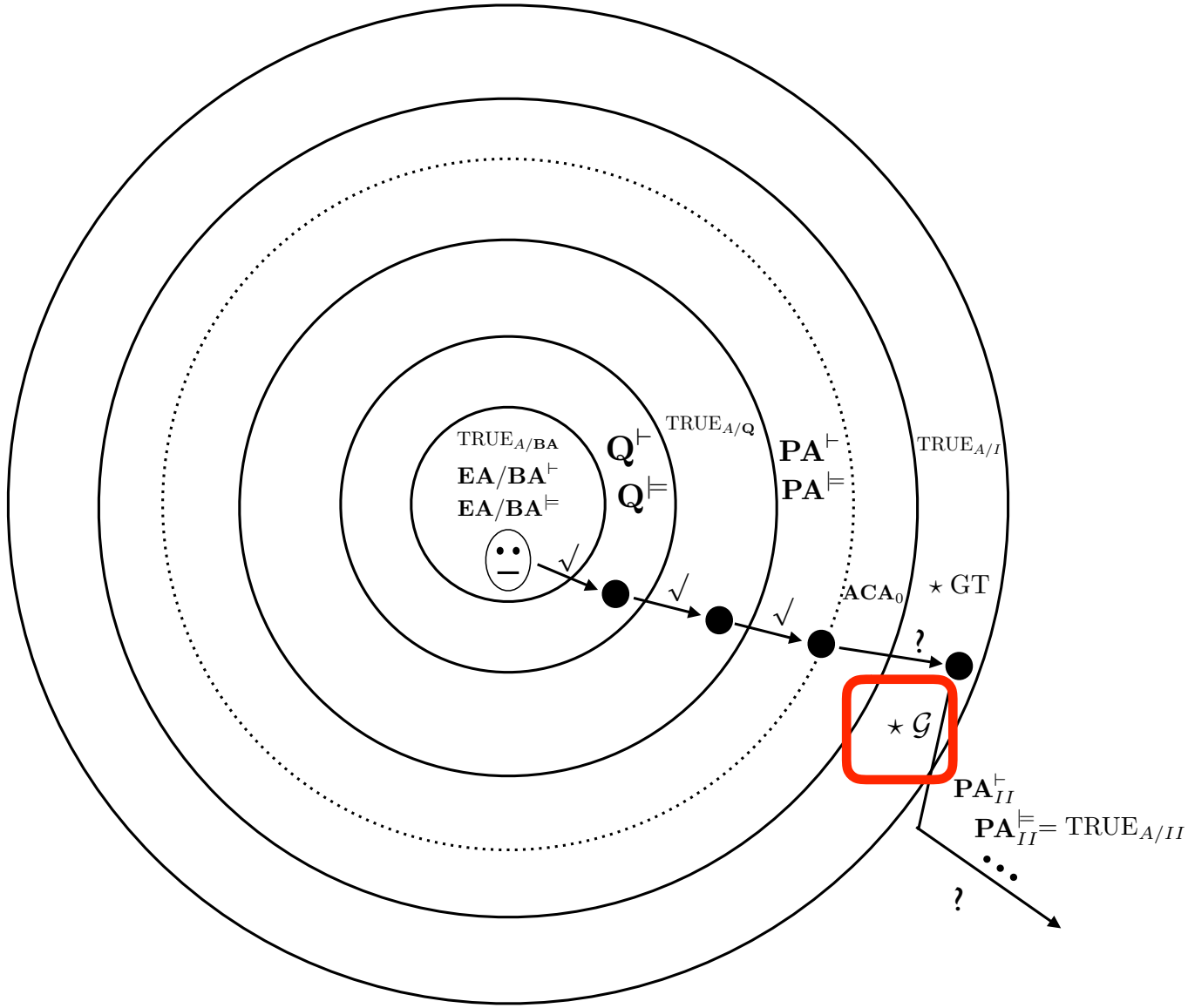
This sequence actually goes to zero!

Astrologic:

Rational Aliens Will be on the Same "Race Track"!



Astrologic:
Rational Aliens Will be on the Same "Race Track"!



Could an AI Ever Match Gödel here?

Actually, thanks to AFOSR:
GI (& GT): “Done” ...

Analogico-Deductive Generation of Gödel's First Incompleteness Theorem from the Liar Paradox

John Licato, Naveen Sundar Govindarajulu, Selmer Bringsjord, Michael Pomeranz, Logan Gittelson
Rensselaer Polytechnic Institute
Troy, NY
{licatj,govinn,selmer,pomerm,gittel}@rpi.edu

Abstract

Gödel's proof of his famous first incompleteness theorem (**G1**) has quite understandably long been a tantalizing target for those wanting to engineer impressively intelligent computational systems. After all, in establishing **G1**, Gödel did something that by any metric must be classified as stunningly intelligent. We observe that it has long been understood that there is some sort of analogical relationship between the Liar Paradox (**LP**) and **G1**, and that Gödel himself appreciated and exploited the relationship. Yet the exact nature of the relationship has hitherto not been uncovered, by which we mean that the following question has not been answered: Given a description of **LP**, and the suspicion that it may somehow be used by a suitably programmed computing machine to find a proof of the incompleteness of Peano Arithmetic, can such a machine, provided this description as input, produce as output a complete and verifiably correct proof of **G1**? In this paper, we summarize engineering that entails an affirmative answer to this question. Our approach uses what we call *analogico-deductive reasoning* (ADR), which combines analogical and deductive reasoning to produce a full deductive proof of **G1** from **LP**. Our engineering uses a form of ADR based on our META-R system, and a connection between the Liar Sentence in **LP** and Gödel's Fixed Point Lemma, from which **G1** follows quickly.

1 Introduction

Gödel's proofs of his incompleteness theorems are among the greatest intellectual achievements of the 20th century. Even armed with the suggestion that the Liar Paradox (**LP**) might somehow be useful as a guide to proving the incompleteness of Peano Arithmetic (**PA**),[1] the level of creativity and philosophical clarity required to actually tie the two concepts together and produce a valid proof is staggering; it certainly should not be controversial to claim that no computational reasoning system can, at present, achieve this sort of feat without significant human assistance.

1.1 Automating the Proof of G1

Prior work devoted to producing computational systems able to prove **G1** have yielded systems able to prove this theorem only when the distance between this result and the starting point is quite small. This for example holds for the first (and certainly seminal) foray; i.e., for [Quaife, 1988], as explained in [Bringsjord, 1998], where it's shown that the proof of **G1**, because the set of premises includes an ingenious human-devised encoding scheme, is very easy—to the point of being at the level of proofs requested from students in introductory mathematical logic classes.

Likewise, [Arnon, 1993] is an exact parallel of the human-devised proof given by [Kleene, 1996]. Finally, in much more recent and truly impressive work by [Sieg and Field, 2005], there is a move to natural-deduction formats, which we applaud—but the machine essentially begins its processing at a point exceedingly close to where it needs to end up. As Sieg and Field concede: "As axioms we take for granted the representability and derivability conditions for the central syntactic notions as well as the diagonal lemma for constructing self-referential sentences." If one takes for granted such things, finding a proof of **G1** is effortless for a computing machine.[2] In sum, while a lot of commendable work has been done to build the foundation for our prospective work, the daunting formal and engineering challenge of producing a computational system able to produce **G1** without clever seeding from a human remains entirely unmet.

2 The Analogico-Deductive Approach

2.1 Conjecture Generation

The problem with the purely deductive method is simply that it does not allow us to come close to the type of model-based reasoning that great thinkers are known to have used. Gödel himself has been described as having a "line of thought [which] seems to move from conjecture to conjecture" [Wang, 1995]. Reasoners in general are known to conjecture through analogy when a straightforward answer

[1] **G1** of course applies to any axiom system meeting the standard conditions (Turing-decidability, representability, consistency), but we tend to refer to **PA** for economization.

[2] A video demonstration of the small-distance process can be found at http://kryten.mm.rpi.edu/GodelI_abstract_in_Slate.mov

Licato, J.; Govindarajulu, N.; Bringsjord, S.; Pomeranz, M.; Gittelson, L. 2013. Analogico-Deductive Generation of Godel's First Incompleteness Theorem from the Liar Paradox. In *Proceedings of IJCAI 2013*. Pdf

Small Steps Toward Hypercomputation via Infinitary Machine Proof Verification and Proof Generation

Naveen Sundar Govindarajulu, John Licato, and Selmer Bringsjord
Department of Computer Science
Department of Cognitive Science
Rensselaer AI & Reasoning Laboratory
govinn@rpi.edu • licatj@rpi.edu • selmer@rpi.edu

Rensselaer Polytechnic Institute
110 8th Street, Troy , NY 12180 USA

Abstract. After setting a context based on two general points (that humans appear to reason in infinitary fashion, and two, that actual hypercomputers aren't currently available to directly model and replicate such infinitary reasoning), we set a humble engineering goal of taking initial steps toward a computing machine that can reason in infinitary fashion. The initial steps consist in our outline of automated proof-verification and proof-discovery techniques for theorems independent of PA that seem to require an understanding and use of infinitary concepts. We specifically focus on proof-discovery techniques that make use of a marriage of analogical and deductive reasoning (which we call *analogico-deductive reasoning*).

A Context: Infinitary Reasoning, Hypercomputation, and Humble Engineering

Bringsjord has repeatedly pointed out the obvious fact that the behavior of formal scientists, taken at face value, involve various infinitary structures and reasoning. (We say "at face value" to simply indicate we don't presuppose some view that denies the reality of infinite entities routinely involved in the formal sciences.) For example, in (Bringsjord & van Heuveln 2003), Bringsjord himself operates as such a scientist in presenting an infinitary paradox which to his knowledge has yet to be solved. And he has argued that apparently infinitary behavior constitutes a grave challenge to AI and the Church-Turing Thesis (e.g., see Bringsjord & Arkoudas 2006, Bringsjord & Zenzen 2003). More generally, Bringsjord conjectures that every human-produced proof of a theorem independent of Peano Arithmetic (PA) will make use of infinitary structures and reasoning, when these structures are taken at face value.[1] We have ourselves designed logico-computational logics for handling infinitary reasoning (e.g., see the treatment of the infinitized wise-man puzzle: Arkoudas & Bringsjord 2005), but this work simply falls back on the human ability to carry out induction on the natural numbers: it doesn't dissect and explain this ability. Finally, it must be admitted by all that there is simply no systematic, comprehensive model or framework anywhere in the formal/computational approach to understanding human knowledge and intelligence that provides a theory about how humans are able to engage with infinitary structures. This is revealed perhaps most clearly when one studies the fruit produced by the part of formal AI devoted to producing discovery systems: such fruit is embarrassingly finitary (e.g., see Shilliday 2009).

Given this context, we are interested in exploring how one might give a machine the ability to reason in infinitary fashion. We are not saying that we in fact have figured out how to give such ability to a computing machine. Our objective here is much more humble and limited: it is to push forward in the *attempt* to engineer a computing machine that has the ability to reason in infinitary fashion. Ultimately, if such an attempt is to succeed, the computing machine in question will presumably be capable of outright hypercomputation. But the fact is that from an engineering perspective, we don't know how to create and harness a hypercomputer. So what we must first try to do, as explained in (Bringsjord & Zenzen 2003), is pursue engineering that initiates the attempt to engineer a hypercomputer, and takes the first few steps. In the present paper, the engineering is aimed specifically at giving a computing machine the ability to, in a limited but well-defined sense, reason in infinitary fashion. Even more specifically, our engineering is aimed at building a machine capable of at least providing a strong case for a result which, in the human sphere, has hitherto required use of infinitary techniques.

[1] A weaker conjecture along the same line has been ventured by Isaacson, and is elegantly discussed by Smith (2007).

Govindarajulu, N.; Licato, J.; Bringsjord, S. 2013. Small Steps Toward Hypercomputation via Infinitary Machine Proof Verification and Proof Generation. In *Proceedings of UCNC 2013*. Pdf

Analogico-Deductive Generation of Gödel's First Incompleteness Theorem from the Liar Paradox

John Licato, Naveen Sundar Govindarajulu, Selmer Bringsjord, Michael Pomeranz, Logan Gittelson
Rensselaer Polytechnic Institute
Troy, NY
{licatj,govinn,selmer,pomerm,gittel}@rpi.edu

Abstract

Gödel's proof of his famous first incompleteness theorem (**G1**) has quite understandably long been a tantalizing target for those wanting to engineer impressively intelligent computational systems. After all, in establishing **G1**, Gödel did something that by any metric must be classified as stunningly intelligent. We observe that it has long been understood that there is some sort of analogical relationship between the Liar Paradox (**LP**) and **G1**, and that Gödel himself appreciated and exploited the relationship. Yet the exact nature of the relationship has hitherto not been uncovered, by which we mean that the following question has not been answered: Given a description of **LP**, and the suspicion that it may somehow be used by a suitably programmed computing machine to find a proof of the incompleteness of Peano Arithmetic, can such a machine, provided this description as input, produce as output a complete and verifiably correct proof of **G1**? In this paper, we summarize engineering that entails an affirmative answer to this question. Our approach uses what we call *analogico-deductive reasoning* (ADR), which combines analogical and deductive reasoning to produce a full deductive proof of **G1** from **LP**. Our engineering uses a form of ADR based on our META-R system, and a connection between the Liar Sentence in **LP** and Gödel's Fixed Point Lemma, from which **G1** follows quickly.

1 Introduction

Gödel's proofs of his incompleteness theorems are among the greatest intellectual achievements of the 20th century. Even armed with the suggestion that the Liar Paradox (**LP**) might somehow be useful as a guide to proving the incompleteness of Peano Arithmetic (**PA**),[1] the level of creativity and philosophical clarity required to actually tie the two concepts together and produce a valid proof is staggering; it certainly should not be controversial to claim that no computational reasoning system can, at present, achieve this sort of feat without significant human assistance.

1.1 Automating the Proof of G1

Prior work devoted to producing computational systems able to prove **G1** have yielded systems able to prove this theorem only when the distance between this result and the starting point is quite small. This for example holds for the first (and certainly seminal) foray; i.e., for [Quaife, 1988], as explained in [Bringsjord, 1998], where it's shown that the proof of **G1**, because the set of premises includes an ingenious human-devised encoding scheme, is very easy—to the point of being at the level of proofs requested from students in introductory mathematical logic classes.

Likewise, [Arnon, 1993] is an exact parallel of the human-devised proof given by [Kleene, 1996]. Finally, in much more recent and truly impressive work by [Sieg and Field, 2005], there is a move to natural-deduction formats, which we applaud—but the machine essentially begins its processing at a point exceedingly close to where it needs to end up. As Sieg and Field concede: "As axioms we take for granted the representability and derivability conditions for the central syntactic notions as well as the diagonal lemma for constructing self-referential sentences." If one takes for granted such things, finding a proof of **G1** is effortless for a computing machine.[2] In sum, while a lot of commendable work has been done to build the foundation for our prospective work, the daunting formal and engineering challenge of producing a computational system able to produce **G1** without clever seeding from a human remains entirely unmet.

2 The Analogico-Deductive Approach

2.1 Conjecture Generation

The problem with the purely deductive method is simply that it does not allow us to come close to the type of model-based reasoning that great thinkers are known to have used. Gödel himself has been described as having a "line of thought [which] seems to move from conjecture to conjecture" [Wang, 1995]. Reasoners in general are known to conjecture through analogy when a straightforward answer

[1] **G1** of course applies to any axiom system meeting the standard conditions (Turing-decidability, representability, consistency), but we tend to refer to **PA** for economization.

[2] A video demonstration of the small-distance process can be found at http://kryten.mm.rpi.edu/GodelI_abstract_in_Slate.mov

Licato, J.; Govindarajulu, N.; Bringsjord, S.; Pomeranz, M.; Gittelson, L. 2013. Analogico-Deductive Generation of Godel's First Incompleteness Theorem from the Liar Paradox. In *Proceedings of IJCAI 2013*. Pdf

Small Steps Toward Hypercomputation via Infinitary Machine Proof Verification and Proof Generation

Naveen Sundar Govindarajulu, John Licato, and Selmer Bringsjord
Department of Computer Science
Department of Cognitive Science
Rensselaer AI & Reasoning Laboratory
govinn@rpi.edu • licatj@rpi.edu • selmer@rpi.edu

Rensselaer Polytechnic Institute
110 8th Street, Troy , NY 12180 USA

Abstract. After setting a context based on two general points (that humans appear to reason in infinitary fashion, and two, that actual hypercomputers aren't currently available to directly model and replicate such infinitary reasoning), we set a humble engineering goal of taking initial steps toward a computing machine that can reason in infinitary fashion. The initial steps consist in our outline of automated proof-verification and proof-discovery techniques for theorems independent of PA that seem to require an understanding and use of infinitary concepts. We specifically focus on proof-discovery techniques that make use of a marriage of analogical and deductive reasoning (which we call *analogico-deductive reasoning*).

A Context: Infinitary Reasoning, Hypercomputation, and Humble Engineering

Bringsjord has repeatedly pointed out the obvious fact that the behavior of formal scientists, taken at face value, involve various infinitary structures and reasoning. (We say "at face value" to simply indicate we don't presuppose some view that denies the reality of infinite entities routinely involved in the formal sciences.) For example, in (Bringsjord & van Heuveln 2003), Bringsjord himself operates as such a scientist in presenting an infinitary paradox which to his knowledge has yet to be solved. And he has argued that apparently infinitary behavior constitutes a grave challenge to AI and the Church-Turing Thesis (e.g., see Bringsjord & Arkoudas 2006, Bringsjord & Zenzen 2003). More generally, Bringsjord conjectures that every human-produced proof of a theorem independent of Peano Arithmetic (PA) will make use of infinitary structures and reasoning, when these structures are taken at face value.[1] We have ourselves designed logico-computational logics for handling infinitary reasoning (e.g., see the treatment of the infinitized wise-man puzzle: Arkoudas & Bringsjord 2005), but this work simply falls back on the human ability to carry out induction on the natural numbers: it doesn't dissect and explain this ability. Finally, it must be admitted by all that there is simply no systematic, comprehensive model or framework anywhere in the formal/computational approach to understanding human knowledge and intelligence that provides a theory about how humans are able to engage with infinitary structures. This is revealed perhaps most clearly when one studies the fruit produced by the part of formal AI devoted to producing discovery systems: such fruit is embarrassingly finitary (e.g., see Shilliday 2009).

Given this context, we are interested in exploring how one might give a machine the ability to reason in infinitary fashion. We are not saying that we in fact have figured out how to give such ability to a computing machine. Our objective here is much more humble and limited: it is to push forward in the *attempt* to engineer a computing machine that has the ability to reason in infinitary fashion. Ultimately, if such an attempt is to succeed, the computing machine in question will presumably be capable of outright hypercomputation. But the fact is that from an engineering perspective, we don't know how to create and harness a hypercomputer. So what we must first try to do, as explained in (Bringsjord & Zenzen 2003), is pursue engineering that initiates the attempt to engineer a hypercomputer, and takes the first few steps. In the present paper, the engineering is aimed specifically at giving a computing machine the ability to, in a limited but well-defined sense, reason in infinitary fashion. Even more specifically, our engineering is aimed at building a machine capable of at least providing a strong case for a result which, in the human sphere, has hitherto required use of infinitary techniques.

[1] A weaker conjecture along the same line has been ventured by Isaacson, and is elegantly discussed by Smith (2007).

Govindarajulu, N.; Licato, J.; Bringsjord, S. 2013. Small Steps Toward Hypercomputation via Infinitary Machine Proof Verification and Proof Generation. In *Proceedings of UCNC 2013*. Pdf

Continuum of Results

Liar Paradox (L)

- Collection of semiformal statements
- Syntax not rigorously defined
- Represents intuitive understanding of problem domain

s = "This statement is a lie"
There is a statement that is neither true nor false.
...

G1

- Completely formal statements
- Syntax very rigorously defined
- Purely mathematical objects: numbers, formal theories, etc.

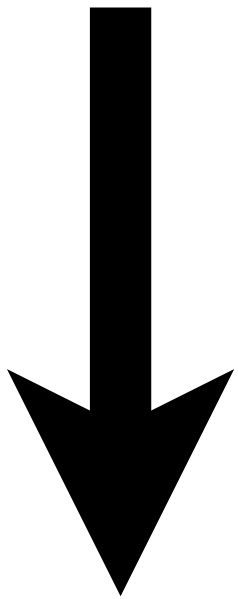
s = ?
$\exists_{\varphi \in L_A} \neg(PA \vdash \varphi) \wedge \neg(PA \vdash \neg\varphi)$
...

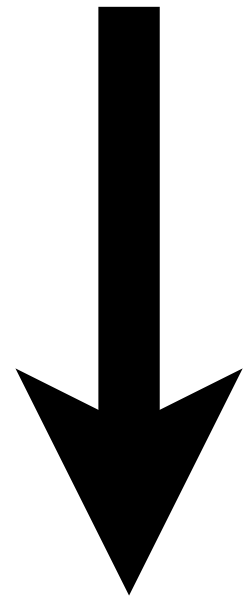
Continuum of Results

Liar Paradox (L)

- Collection of semiformal statements
- Syntax not rigorously defined
- Represents intuitive understanding of problem domain

s = "This statement is a lie"
There is a statement that is neither true nor false.
...





G1

- Completely formal statements
- Syntax very rigorously defined
- Purely mathematical objects: numbers, formal theories, etc.

s = ?
$\exists_{\varphi \in L_A} \neg(PA \vdash \varphi) \wedge \neg(PA \vdash \neg\varphi)$
...

Continuum of Results

Liar Paradox (L)

- Collection of semiformal statements
- Syntax not rigorously defined
- Represents intuitive understanding of problem domain

s = "This statement is a lie"
There is a statement that is neither true nor false.
...

S

- Semiformal statements (but more formal than L)
- Syntax somewhat rigorously defined
- Somewhat intuitive; deals with stories of reasoners and utterances made by inhabitants of an island

$s = \neg Bs$
$\exists_p \neg Bp \wedge \neg B\neg p$
...

G1

- Completely formal statements
- Syntax very rigorously defined
- Purely mathematical objects: numbers, formal theories, etc.

$s = ?$
$\exists_{\varphi \in L_A} \neg(PA \vdash \varphi) \wedge \neg(PA \vdash \neg\varphi)$
...

Continuum of Results

Liar Paradox (L)

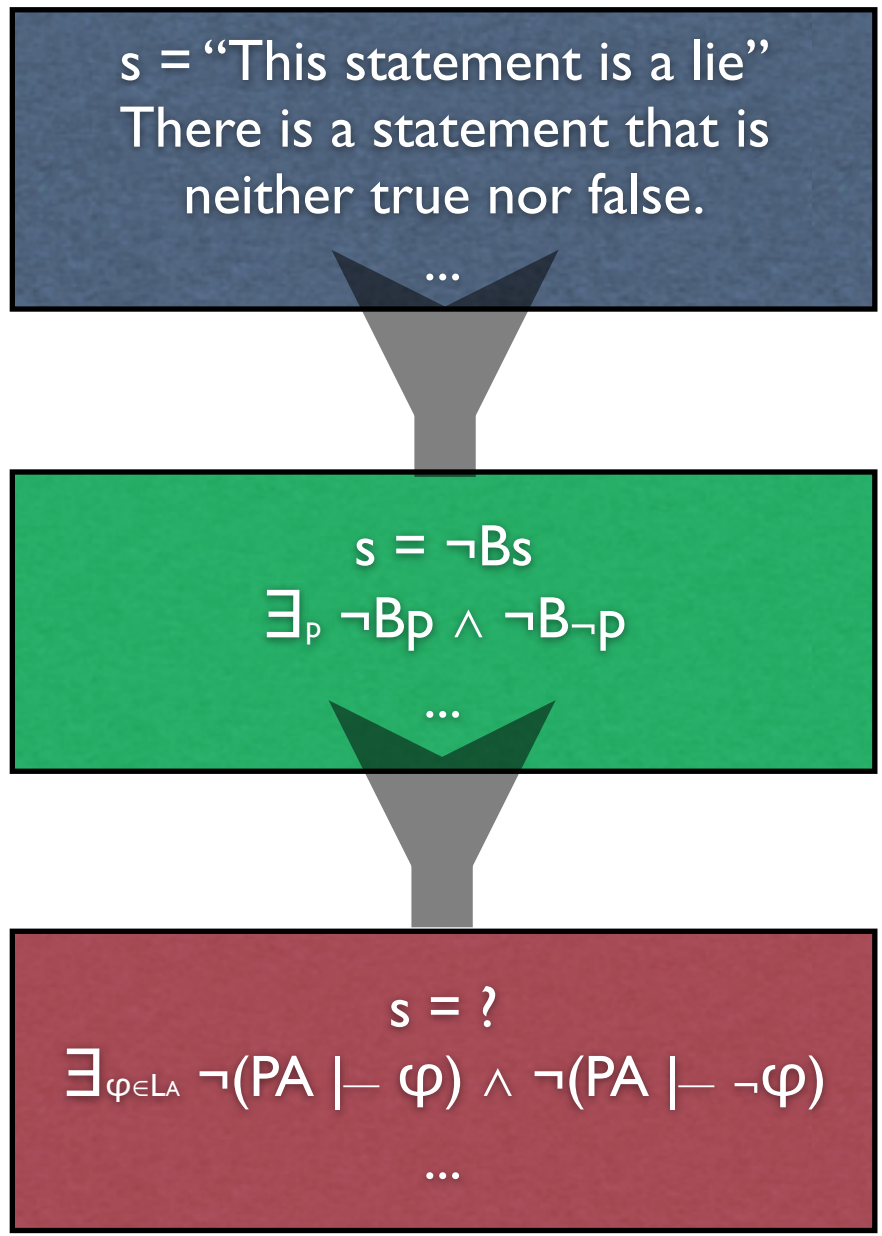
- Collection of semiformal statements
- Syntax not rigorously defined
- Represents intuitive understanding of problem domain

S

- Semiformal statements (but more formal than L)
- Syntax somewhat rigorously defined
- Somewhat intuitive; deals with stories of reasoners and utterances made by inhabitants of an island

G1

- Completely formal statements
- Syntax very rigorously defined
- Purely mathematical objects: numbers, formal theories, etc.



Continuum of Results

Liar Paradox (L)

- Collection of semiformal statements
- Syntax not rigorously defined
- Represents intuitive understanding of problem domain

S

- Semiformal statements (but more formal than L)
- Syntax somewhat rigorously defined
- Somewhat intuitive; deals with stories of reasoners and utterances made by inhabitants of an island

G1

- Completely formal statements
- Syntax very rigorously defined
- Purely mathematical objects: numbers, formal theories, etc.

s = "This statement is a lie"
There is a statement that is neither true nor false.
...

$s = \neg Bs$
$\exists_p \neg Bp \wedge \neg B\neg p$
...

$s = ?$
$\exists_{\varphi \in L_A} \neg(PA \vdash \varphi) \wedge \neg(PA \vdash \neg\varphi)$
...

Continuum of Results

Liar Paradox (L)

- Collection of semiformal statements
- Syntax not rigorously defined
- Represents intuitive understanding of problem domain

s = "This statement is a lie"
There is a statement that is neither true nor false.
...

S

- Semiformal statements (but more formal than L)
- Syntax somewhat rigorously defined
- Somewhat intuitive; deals with stories of reasoners and utterances made by inhabitants of an island

"Done"

$s = \neg Bs$
$\exists_p \neg Bp \wedge \neg B\neg p$
...

GI

- Completely formal statements
- Syntax very rigorously defined
- Purely mathematical objects: numbers, formal theories, etc.

s = ?
$\exists_{\varphi \in L_A} \neg(PA \vdash \varphi) \wedge \neg(PA \vdash \neg\varphi)$
...

slutten