

“The Wager,” Overview of the Book, and Gödel’s Completeness Theorem

Selmer Bringsjord

Rensselaer AI & Reasoning (RAIR) Lab
Department of Cognitive Science
Department of Computer Science
Lally School of Management & Technology
Rensselaer Polytechnic Institute (RPI)
Troy, New York 12180 USA

4/6/2023



“The Wager,” Overview of the Book, and Gödel’s Completeness Theorem

Selmer Bringsjord

Rensselaer AI & Reasoning (RAIR) Lab
Department of Cognitive Science
Department of Computer Science
Lally School of Management & Technology
Rensselaer Polytechnic Institute (RPI)
Troy, New York 12180 USA

4/6/2023

Note: This is a version of coverage of Gödel’s Completeness Theorem designed for those who’ve had at least one standard/standard-paced university-level course in formal logic.



OXFORD
UNIVERSITY PRESS



Background Context ...

Gödel's Great Theorems (OUP)


by Selmer Bringsjord

- Introduction (“The Wager”)
- Brief Preliminaries (elementary discrete math, incl. ZOL, 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 (elementary discrete math, incl. ZOL, 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 (elementary discrete math, incl. ZOL, 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 (elementary discrete math, incl. ZOL, 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?



Some Timeline Points



Some Timeline Points

1906 Brünn, Austria-Hungary



Some Timeline Points

1923 Vienna

1906 Brünn, Austria-Hungary



Some Timeline Points

Undergrad in seminar by Schlick

1923 Vienna

1906 Brünn, Austria-Hungary



Some Timeline Points

1929 Doctoral Dissertation: Proof of Completeness Theorem
Undergrad in seminar by Schlick

1923 Vienna

1906 Brünn, Austria-Hungary



Some Timeline Points

1930 Announces (First) *Incompleteness* Theorem
1929 Doctoral Dissertation: Proof of Completeness Theorem
Undergrad in seminar by Schlick

1923 Vienna

1906 Brünn, Austria-Hungary



Some Timeline Points

1933 Hitler comes to power.

1930 Announces (First) *Incompleteness* Theorem

1929 Doctoral Dissertation: Proof of Completeness Theorem

Undergrad in seminar by Schlick

1923 Vienna

1906 Brünn, Austria-Hungary



Some Timeline Points

1940 Back to USA, for good.

1933 Hitler comes to power.

1930 Announces (First) *Incompleteness* Theorem

1929 Doctoral Dissertation: Proof of Completeness Theorem

Undergrad in seminar by Schlick

1923 Vienna

1906 Brünn, Austria-Hungary



Some Timeline Points



1940 Back to USA, for good.

1933 Hitler comes to power.

1930 Announces (First) *Incompleteness* 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.

1933 Hitler comes to power.

1930 Announces (First) *Incompleteness* 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; advisor dies

1933 Hitler comes to power.

1930 Announces (First) *Incompleteness* 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; advisor dies

1933 Hitler comes to power.

1930 Announces (First) *Incompleteness* 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; advisor dies

1933 Hitler comes to power.

1930 Announces (First) *Incompleteness* 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; advisor dies

1933 Hitler comes to power.

1930 Announces (First) *Incompleteness* Theorem

1929 Doctoral Dissertation: Proof of Completeness Theorem

Undergrad in seminar by Schlick

1923 Vienna

1906 Brünn, Austria-Hungary

“I have proved that syntax and semantics are fundamentally the same.”



**Preliminaries:
Propositional Calculus &
First-Order Logic**

...

Actually ...

$$\mathcal{L}_0 < \mathcal{L}_1 < \mathcal{L}_2 < \mathcal{L}_3 \dots$$

Actually ...

Second-order logic.

Third-order logic, which Gödel used for his “God Theorem.”

$\mathcal{L}_0 < \mathcal{L}_1 < \mathcal{L}_2 < \mathcal{L}_3 \dots$

Zero-order logic; subsumes the propositional calculus.

First-order logic; this is what the Completeness Theorem is about: i.e., this logic is complete.



Actually ...

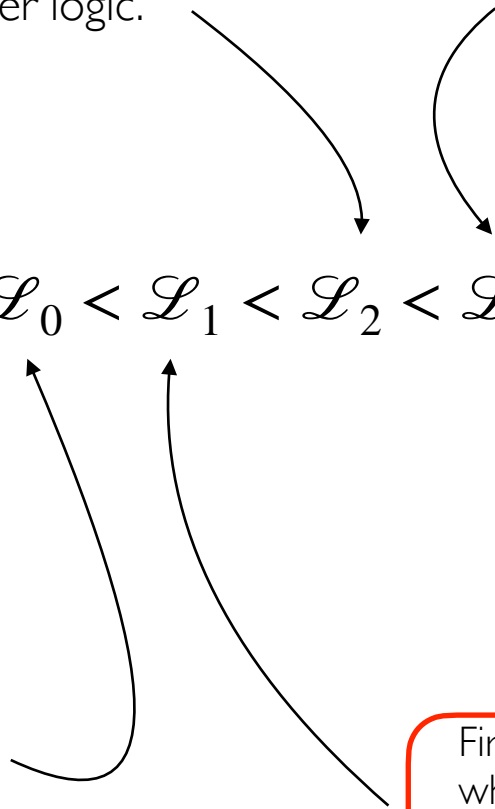
Second-order logic.

Third-order logic, which Gödel used for his “God Theorem.”

$\mathcal{L}_0 < \mathcal{L}_1 < \mathcal{L}_2 < \mathcal{L}_3 \dots$

Zero-order logic; subsumes the propositional calculus.

First-order logic; this is what the Completeness Theorem is about: i.e., this logic is complete.



Actually ...

This logic is *not* complete.

Second-order logic.

Third-order logic, which Gödel used for his “God Theorem.”

$\mathcal{L}_0 < \mathcal{L}_1 < \mathcal{L}_2 < \mathcal{L}_3 \dots$

Zero-order logic; subsumes the propositional calculus.

First-order logic; this is what the Completeness Theorem is about: i.e., this logic is complete.



R&W's Axiomatization of the Propositional Calculus

$$A1 \quad (\phi \vee \phi) \rightarrow \phi$$

$$A2 \quad \phi \rightarrow (\phi \vee \psi)$$

$$A3 \quad (\phi \vee \psi) \rightarrow (\psi \vee \phi)$$

$$A4 \quad (\psi \rightarrow \chi) \rightarrow ((\phi \vee \psi) \rightarrow (\phi \vee \chi))$$



R&W's Axiomatization of the Propositional Calculus

$$A1 \quad (\phi \vee \phi) \rightarrow \phi$$

$$A2 \quad \phi \rightarrow (\phi \vee \psi)$$

$$A3 \quad (\phi \vee \psi) \rightarrow (\psi \vee \phi)$$

$$A4 \quad (\psi \rightarrow \chi) \rightarrow ((\phi \vee \psi) \rightarrow (\phi \vee \chi))$$

All instances of these schemata are true no matter what the input (true or false). (Agreed?) And indeed every single formula in the propositional calculus that is true no matter what the permutation (as shown in a truth table, e.g.), can be proved (somehow) from these four axioms (using any standard collection of inference schemata). This, Gödel (& later, Newell & Simon, when modern AI was born!) knew, and could use.



R&W's Axiomatization of the Propositional Calculus

A1	$(\phi \vee \phi) \rightarrow \phi$	<code>(if (or \phi \phi) \phi)</code>
A2	$\phi \rightarrow (\phi \vee \psi)$	<code>(if \phi (or \phi \psi))</code>
A3	$(\phi \vee \psi) \rightarrow (\psi \vee \phi)$	<code>(if (or \phi \psi) (or \psi \phi))</code>
A4	$(\psi \rightarrow \chi) \rightarrow ((\phi \vee \psi) \rightarrow (\phi \vee \chi))$	<code>(if (if \psi \chi) (if (or \phi \psi) (or \phi \chi)))</code>

All instances of these schemata are true no matter what the input (true or false). (Agreed?) And indeed every single formula in the propositional calculus that is true no matter what the permutation (as shown in a truth table, e.g.), can be proved (somehow) from these four axioms (using any standard collection of inference schemata). This, Gödel (& later, Newell & Simon, when modern AI was born!) knew, and could use.

Exercise I:

Verify that these are true-no-matter what in a truth tree in HyperSlate[®];
then prove using our rules for the prop. calc.; or perhaps better yet,
have the oracle prove in HyperSlate[®].

$$(\phi \wedge \psi) \rightarrow (\psi \vee \chi)$$

$$\phi \rightarrow (\psi \rightarrow \phi)$$

Exercise 1:

Verify that these are true-no-matter what in a truth tree;
then prove using our rules for the prop. calc.

$$(\phi \wedge \psi) \rightarrow (\psi \vee \chi)$$

Exercise I:

Verify that these are true-no-matter what in a truth tree;
then prove using our rules for the prop. calc.



$$(\phi \wedge \psi) \rightarrow (\psi \vee \chi)$$

Truth Tree showing this formula true no matter what the inputs.

Exercise I:

Verify that these are true-no-matter what in a truth tree;
then prove using our rules for the prop. calc.



$$(\phi \wedge \psi) \rightarrow (\psi \vee \chi)$$

Truth Tree showing this formula true no matter what the inputs.

Proof:

Exercise I:

Verify that these are true-no-matter what in a truth tree;
then prove using our rules for the prop. calc.



$$(\phi \wedge \psi) \rightarrow (\psi \vee \chi)$$

Truth Tree showing this formula true no matter what the inputs.

Proof:

Exercise I:

Verify that these are t
then prove using

what in a truth tree;
for the prop. calc.

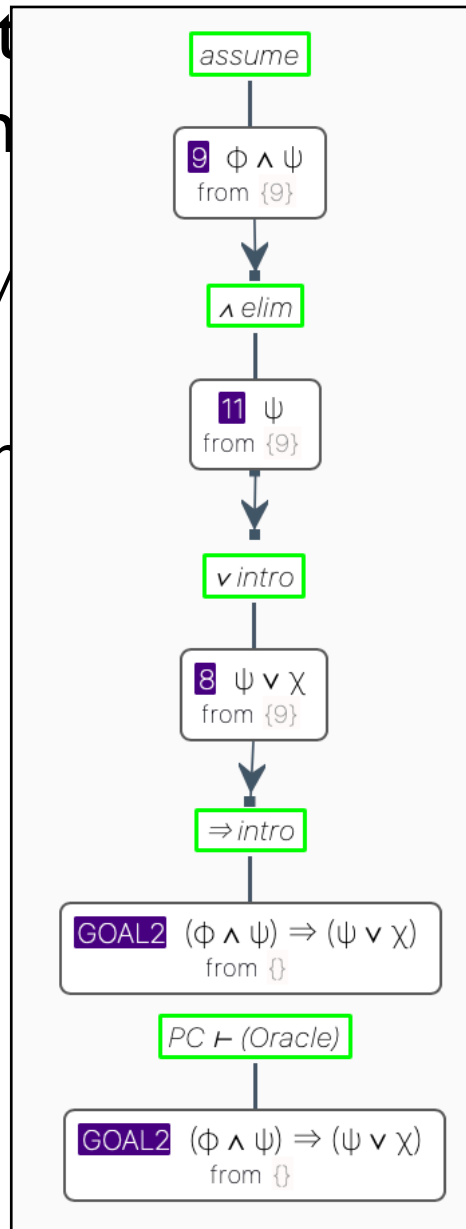


$(\phi \wedge \psi) \Rightarrow (\psi \vee \chi)$

Truth Tree showing th

no matter what the inputs.

Proof:



Exercise I:

Verify that these are t
then prove using

what in a truth tree;
for the prop. calc.

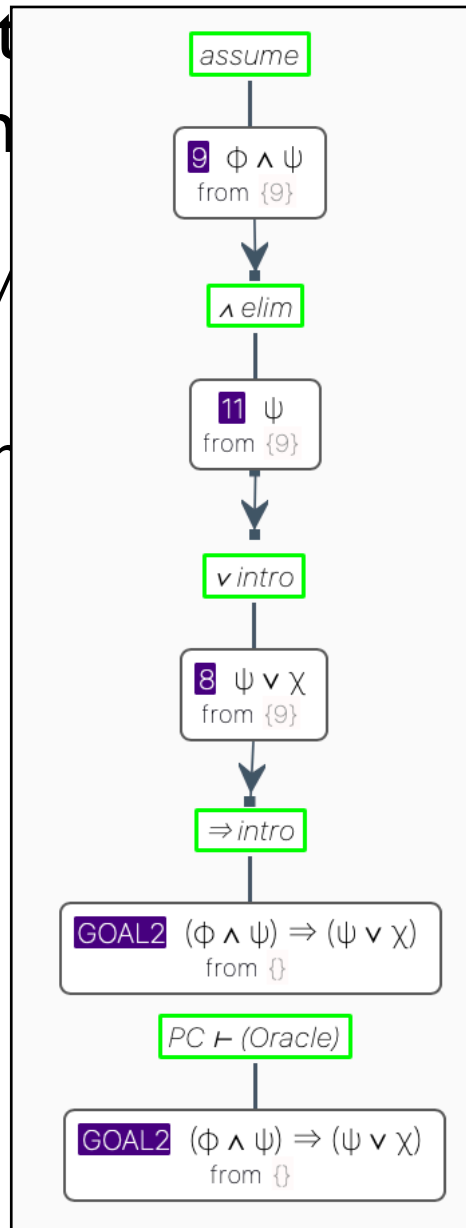


$(\phi \wedge \psi) \Rightarrow (\psi \vee \chi)$

Truth Tree showing th

no matter what the inputs.

Proof:



resolution-based!

Exercise I:

Verify that these are t
then prove using

what in a truth tree;
for the prop. calc.

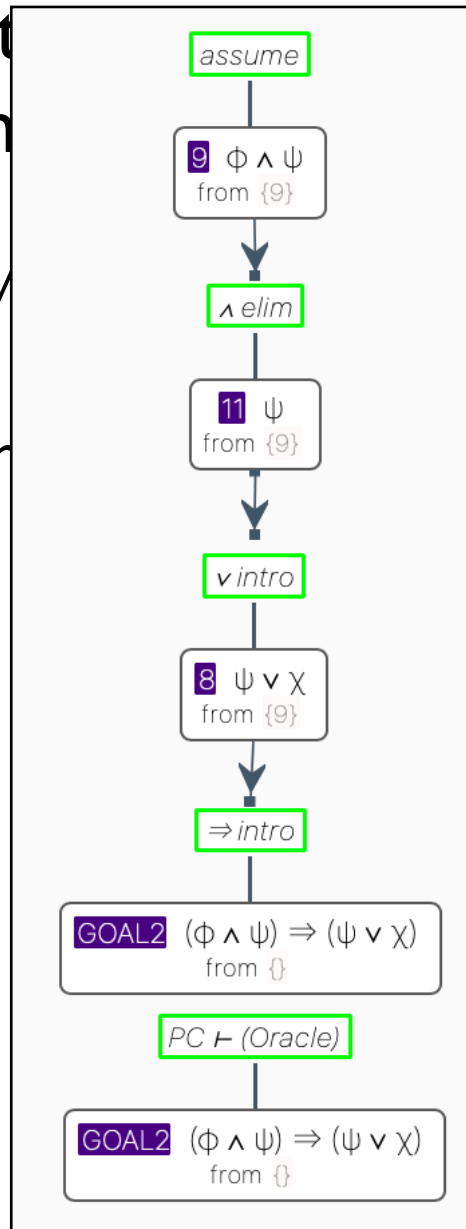


Truth Tree showing th

$(\phi \wedge \psi) \Rightarrow (\psi \vee \chi)$

no matter what the inputs.

Proof:



resolution-based!



As HyperSlate[®] Tutorial

The screenshot shows a web browser window with the URL `rpi.logicamodernapproach.com`. The browser's address bar and tabs are visible. The HyperSlate interface includes a toolbar with icons for home, list, add, save, undo, redo, and Bezier. The main workspace displays a logical proof diagram with the following components:

- AXIOM1**: $(\phi \vee \phi) \Rightarrow \phi$ from {AXIOM1}. Above it is a green box labeled "assume".
- AXIOM2**: $\phi \Rightarrow (\phi \vee \psi)$ from {AXIOM2}. Above it is a green box labeled "assume".
- AXIOM3**: $(\phi \vee \psi) \Rightarrow (\psi \vee \phi)$ from {AXIOM3}. Above it is a green box labeled "assume".
- AXIOM4**: $(\psi \Rightarrow \chi) \Rightarrow ((\phi \vee \psi) \Rightarrow (\phi \vee \chi))$ from {AXIOM4}. Above it is a green box labeled "assume".
- GOAL1**: $\phi \vee \neg\phi$ from {}. Above it is a green box labeled "PC \vdash (Oracle)".
- GOAL2**: $(\phi \wedge \psi) \Rightarrow (\psi \vee \chi)$ from {}. Above it is a green box labeled "PC \vdash (Oracle)".

The diagram shows a vertical chain of four axiom boxes on the left, with a goal box to their right. A second goal box is positioned further down and to the right. The interface also shows a status bar at the bottom indicating "Saved with 39 symbols."

As HyperSlate[®] Tutorial

The screenshot shows a Safari browser window displaying the HyperSlate web application. The address bar shows the URL `rpi.logicamodernapproach.com`. The browser's menu bar includes "Safari", "File", "Edit", "View", "History", "Bookmarks", "Window", and "Help". The page title is "RussellWhiteheadPropCalcAx [PROPOSITIONAL-CALCULUS]: Saved with 39 symbols." The interface features a toolbar with icons for home, list, add, save, undo, redo, and Bezier. The main workspace contains a logical proof diagram with the following components:

- AXIOM1**: $(\phi \vee \phi) \Rightarrow \phi$ from {AXIOM1}. Above it is a green box labeled "assume".
- AXIOM2**: $\phi \Rightarrow (\phi \vee \psi)$ from {AXIOM2}. Above it is a green box labeled "assume".
- AXIOM3**: $(\phi \vee \psi) \Rightarrow (\psi \vee \phi)$ from {AXIOM3}. Above it is a green box labeled "assume".
- AXIOM4**: $(\psi \Rightarrow \chi) \Rightarrow ((\phi \vee \psi) \Rightarrow (\phi \vee \chi))$ from {AXIOM4}. Above it is a green box labeled "assume".
- GOAL1**: $\phi \vee \neg\phi$ from {}. Above it is a green box labeled "PC ⊢ (Oracle)".
- GOAL2**: $(\phi \wedge \psi) \Rightarrow (\psi \vee \chi)$ from {}. Above it is a green box labeled "PC ⊢ (Oracle)".

The diagram shows a vertical chain of four axiom boxes on the left, with two goal boxes on the right. A mouse cursor is pointing at the "GOAL1" box.

The Grammar of \mathcal{L}_0 = the Pure Predicate Calculus

Formula \Rightarrow *AtomicFormula*
| (*Formula* *Connective* *Formula*)
| \neg *Formula*

AtomicFormula \Rightarrow (*Predicate* *Term*₁ ... *Term*_k)
| (*Term* = *Term*)

Term \Rightarrow (*Function* *Term*₁ ... *Term*_k)
| *Constant*

Connective \Rightarrow \wedge | \vee | \rightarrow | \leftrightarrow

Predicate \Rightarrow P_1 | P_2 | P_3 ...

Constant \Rightarrow c_1 | c_2 | c_3 ...

Function \Rightarrow f_1 | f_2 | f_3 ...

Some Simple Examples

Formula \Rightarrow *AtomicFormula*
 | *(Formula* *Connective* *Formula)*
 | \neg *Formula*

Sally likes Bill.
 (**Likes** sally bill)

AtomicFormula \Rightarrow *(Predicate* *Term*₁ ... *Term*_k)
 | *(Term = Term)*

Term \Rightarrow *(Function* *Term*₁ ... *Term*_k)
 | *Constant*

Sally likes Bill and Bill likes Sally.

Sally likes Bill's mother.

Sally likes Bill only if Bill's mother is tall.

Matilda is Bill's super-smart mother.

5 plus 5 equals the number 10.

Connective \Rightarrow \wedge | \vee | \rightarrow | \leftrightarrow

Predicate \Rightarrow P_1 | P_2 | P_3 ...

Constant \Rightarrow c_1 | c_2 | c_3 ...

Function \Rightarrow f_1 | f_2 | f_3 ...

Lexicon

...

Can Roger be counted upon to declare: "Yes that sentence is okay!" whenever it's conforms to this grammar?

Some Simple Examples

Formula ⇒ *AtomicFormula*
 | *(Formula Connective Formula)*
 | \neg *Formula*

Sally likes Bill.
 (Likes sally bill)

AtomicFormula ⇒ *(Predicate Term₁ ... Term_k)*
 | *(Term = Term)*

Term ⇒ *(Function Term₁ ... Term_k)*
 | *Constant*

Sally likes Bill and Bill likes Sally.

Sally likes Bill's mother.

Sally likes Bill only if Bill's mother is tall.

Matilda is Bill's super-smart mother.

5 plus 5 equals the number 10.

Connective ⇒ \wedge | \vee | \rightarrow | \leftrightarrow

Likes
Predicate ⇒ P_1 | P_2 | P_3 ...
Constant ⇒ c_1 | c_2 | c_3 ...
Function ⇒ f_1 | f_2 | f_3 ...

Lexicon

...

Can Roger be counted upon to declare: "Yes that sentence is okay!" whenever it's conforms to this grammar?

Slightly More Complicated Examples

Formula \Rightarrow *AtomicFormula*
| (*Formula* *Connective* *Formula*)
| \neg *Formula*

AtomicFormula \Rightarrow (*Predicate* *Term*₁ ... *Term*_k)
| (*Term* = *Term*)

Term \Rightarrow (*Function* *Term*₁ ... *Term*_k)
| *Constant*

Connective \Rightarrow \wedge | \vee | \rightarrow | \leftrightarrow

Predicate \Rightarrow P_1 | P_2 | P_3 ...

Constant \Rightarrow c_1 | c_2 | c_3 ...

Function \Rightarrow f_1 | f_2 | f_3 ...

Slightly More Complicated Examples

Formula \Rightarrow *AtomicFormula*
| (*Formula* *Connective* *Formula*)
| \neg *Formula*

AtomicFormula \Rightarrow (*Predicate* *Term*₁ ... *Term*_k)
| (*Term* = *Term*)

Term \Rightarrow (*Function* *Term*₁ ... *Term*_k)
| *Constant*

Connective \Rightarrow \wedge | \vee | \rightarrow | \leftrightarrow

Predicate \Rightarrow P_1 | P_2 | P_3 ...

Constant \Rightarrow c_1 | c_2 | c_3 ...

Function \Rightarrow f_1 | f_2 | f_3 ...

If Sally likes Bill then Sally likes Bill.

Slightly More Complicated Examples

Formula \Rightarrow *AtomicFormula*
| (*Formula* *Connective* *Formula*)
| \neg *Formula*

AtomicFormula \Rightarrow (*Predicate* *Term*₁ ... *Term*_k)
| (*Term* = *Term*)

Term \Rightarrow (*Function* *Term*₁ ... *Term*_k)
| *Constant*

Connective \Rightarrow \wedge | \vee | \rightarrow | \leftrightarrow

Predicate \Rightarrow P_1 | P_2 | P_3 ...

Constant \Rightarrow c_1 | c_2 | c_3 ...

Function \Rightarrow f_1 | f_2 | f_3 ...

If Sally likes Bill then Sally likes Bill.

Sally likes Bill's mother, or not.

Slightly More Complicated Examples

Formula \Rightarrow *AtomicFormula*
| (*Formula* *Connective* *Formula*)
| \neg *Formula*

AtomicFormula \Rightarrow (*Predicate* *Term*₁ ... *Term*_k)
| (*Term* = *Term*)

Term \Rightarrow (*Function* *Term*₁ ... *Term*_k)
| *Constant*

Connective \Rightarrow \wedge | \vee | \rightarrow | \leftrightarrow

Predicate \Rightarrow P_1 | P_2 | P_3 ...

Constant \Rightarrow c_1 | c_2 | c_3 ...

Function \Rightarrow f_1 | f_2 | f_3 ...

If Sally likes Bill then Sally likes Bill.

Sally likes Bill's mother, or not.

Sally likes Bill and Bill likes Jane, only if Bill likes Jane.

Slightly More Complicated Examples

Formula \Rightarrow *AtomicFormula*
| (*Formula* *Connective* *Formula*)
| \neg *Formula*

AtomicFormula \Rightarrow (*Predicate* *Term*₁ ... *Term*_k)
| (*Term* = *Term*)

Term \Rightarrow (*Function* *Term*₁ ... *Term*_k)
| *Constant*

Connective \Rightarrow \wedge | \vee | \rightarrow | \leftrightarrow

Predicate \Rightarrow P_1 | P_2 | P_3 ...

Constant \Rightarrow c_1 | c_2 | c_3 ...

Function \Rightarrow f_1 | f_2 | f_3 ...

If Sally likes Bill then Sally likes Bill.

Sally likes Bill's mother, or not.

Sally likes Bill and Bill likes Jane, only if Bill likes Jane.

Bill's smart mother is a mother.

Slightly More Complicated Examples

Formula \Rightarrow *AtomicFormula*
| (*Formula* *Connective* *Formula*)
| \neg *Formula*

AtomicFormula \Rightarrow (*Predicate* *Term*₁ ... *Term*_k)
| (*Term* = *Term*)

Term \Rightarrow (*Function* *Term*₁ ... *Term*_k)
| *Constant*

Connective \Rightarrow \wedge | \vee | \rightarrow | \leftrightarrow

Predicate \Rightarrow P_1 | P_2 | P_3 ...

Constant \Rightarrow c_1 | c_2 | c_3 ...

Function \Rightarrow f_1 | f_2 | f_3 ...

If Sally likes Bill then Sally likes Bill.

Sally likes Bill's mother, or not.

Sally likes Bill and Bill likes Jane, only if Bill likes Jane.

Bill's smart mother is a mother.

...

Slightly More Complicated Examples

Formula \Rightarrow *AtomicFormula*
 | *(Formula Connective Formula)*
 | \neg *Formula*

AtomicFormula \Rightarrow *(Predicate Term₁ ... Term_k)*
 | *(Term = Term)*

Term \Rightarrow *(Function Term₁ ... Term_k)*
 | *Constant*

Connective \Rightarrow \wedge | \vee | \rightarrow | \leftrightarrow

Predicate \Rightarrow P_1 | P_2 | P_3 ...
Constant \Rightarrow c_1 | c_2 | c_3 ...
Function \Rightarrow f_1 | f_2 | f_3 ...

If Sally likes Bill then Sally likes Bill.

Sally likes Bill's mother, or not.

Sally likes Bill and Bill likes Jane, only if Bill likes Jane.

Bill's smart mother is a mother.

...

These are all true, yes; but can they be proved?!

Slightly More Complicated Examples

Formula \Rightarrow *AtomicFormula*
 | *(Formula Connective Formula)*
 | \neg *Formula*

AtomicFormula \Rightarrow *(Predicate Term₁ ... Term_k)*
 | *(Term = Term)*

Term \Rightarrow *(Function Term₁ ... Term_k)*
 | *Constant*

Connective \Rightarrow \wedge | \vee | \rightarrow | \leftrightarrow

Predicate \Rightarrow P_1 | P_2 | P_3 ...
Constant \Rightarrow c_1 | c_2 | c_3 ...
Function \Rightarrow f_1 | f_2 | f_3 ...

If Sally likes Bill then Sally likes Bill.

Sally likes Bill's mother, or not.

Sally likes Bill and Bill likes Jane, only if Bill likes Jane.

Bill's smart mother is a mother.

...

These are all true, yes; but can they be proved?!

Yes!

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

there exists at least one thing x such that ...

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

there exists at least one thing x such that ...

for all x , it's the case that ...

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

$\exists x$. . . there exists at least one thing x such that ...

for all x , it's the case that ...

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

$\exists x \dots$ there exists at least one thing x such that ...

$\forall x \dots$ for all x , it's the case that ...

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

$\exists x$. . . there exists at least one thing x such that . . .

$\forall x$. . . for all x , it's the case that . . .

$$\lim_{x \rightarrow a} f(x) = L$$

iff

$$\forall \epsilon (\epsilon > 0 \rightarrow \exists \delta (\delta > 0 \wedge \forall x (d(x, a) < \delta \rightarrow d(f(x), L) < \epsilon))$$

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

$\exists x \dots$ there exists at least one thing x such that ...

$\forall x \dots$ for all x , it's the case that ...

$$\lim_{x \rightarrow a} f(x) = L$$

```
(forall (\epsilon) (if (> \epsilon 0)
  (exists (\delta) (and (> \delta 0)
    (forall (x) (if (< (dist x a) \delta)
      (< (dist (f x) L) \epsilon)))))))
```

$$\forall \epsilon (\epsilon > 0 \rightarrow \exists \delta (\delta > 0 \wedge \forall x (d(x, a) < \delta \rightarrow d(f(x), L) < \epsilon)))$$

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

$\exists x \dots$ there exists at least one thing x such that ...

$\forall x \dots$ for all x , it's the case that ...

$$\lim_{x \rightarrow a} f(x) = L$$

```
(forall (\epsilon) (if (> \epsilon 0)
  (exists (\delta) (and (> \delta 0)
    (forall (x) (if (< (dist x a) \delta)
      (< (dist (f x) L) \epsilon)))))))
```

$$\forall \epsilon (\epsilon > 0 \rightarrow \exists \delta (\delta > 0 \wedge \forall x (d(x, a) < \delta \rightarrow d(f(x), L) < \epsilon)))$$

Every natural number is greater than or equal to zero.

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

$\exists x \dots$ there exists at least one thing x such that ...

$\forall x \dots$ for all x , it's the case that ...

$$\lim_{x \rightarrow a} f(x) = L$$

```
(forall (\epsilon) (if (> \epsilon 0)
  (exists (\delta) (and (> \delta 0)
    (forall (x) (if (< (dist x a) \delta)
      (< (dist (f x) L) \epsilon)))))))
```

$$\forall \epsilon (\epsilon > 0 \rightarrow \exists \delta (\delta > 0 \wedge \forall x (d(x, a) < \delta \rightarrow d(f(x), L) < \epsilon)))$$

Every natural number is greater than or equal to zero.

$$\forall x (x \geq 0)$$

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

$\exists x \dots$ there exists at least one thing x such that ...

$\forall x \dots$ for all x , it's the case that ...

$$\lim_{x \rightarrow a} f(x) = L$$

```
(forall (\epsilon) (if (> \epsilon 0)
  (exists (\delta) (and (> \delta 0)
    (forall (x) (if (< (dist x a) \delta)
      (< (dist (f x) L) \epsilon))))))))
```

$$\forall \epsilon (\epsilon > 0 \rightarrow \exists \delta (\delta > 0 \wedge \forall x (d(x, a) < \delta \rightarrow d(f(x), L) < \epsilon)))$$

Every natural number is greater than or equal to zero.

$$\forall x (x \geq 0)$$

There's a positive integer greater than any positive integer.

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

$\exists x \dots$ there exists at least one thing x such that ...

$\forall x \dots$ for all x , it's the case that ...

$$\lim_{x \rightarrow a} f(x) = L$$

```
(forall (\epsilon) (if (> \epsilon 0)
  (exists (\delta) (and (> \delta 0)
    (forall (x) (if (< (dist x a) \delta)
      (< (dist (f x) L) \epsilon)))))))
```

$$\forall \epsilon (\epsilon > 0 \rightarrow \exists \delta (\delta > 0 \wedge \forall x (d(x, a) < \delta \rightarrow d(f(x), L) < \epsilon)))$$

Every natural number is greater than or equal to zero.

$$\forall x (x \geq 0)$$

There's a positive integer greater than any positive integer.

$$\exists x \forall y (y < x)$$

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

$\exists x \dots$ there exists at least one thing x such that ...

$\forall x \dots$ for all x , it's the case that ...

$$\lim_{x \rightarrow a} f(x) = L$$

```
(forall (\epsilon) (if (> \epsilon 0)
  (exists (\delta) (and (> \delta 0)
    (forall (x) (if (< (dist x a) \delta)
      (< (dist (f x) L) \epsilon)))))))
```

$$\forall \epsilon (\epsilon > 0 \rightarrow \exists \delta (\delta > 0 \wedge \forall x (d(x, a) < \delta \rightarrow d(f(x), L) < \epsilon)))$$

Every natural number is greater than or equal to zero.

$$\forall x (x \geq 0)$$

$$\exists x \forall y (y < x)$$

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

$\exists x \dots$ there exists at least one thing x such that ...

$\forall x \dots$ for all x , it's the case that ...

$$\lim_{x \rightarrow a} f(x) = L$$

```
(forall (\epsilon) (if (> \epsilon 0)
  (exists (\delta) (and (> \delta 0)
    (forall (x) (if (< (dist x a) \delta)
      (< (dist (f x) L) \epsilon)))))))
```

$$\forall \epsilon (\epsilon > 0 \rightarrow \exists \delta (\delta > 0 \wedge \forall x (d(x, a) < \delta \rightarrow d(f(x), L) < \epsilon)))$$

Every natural number is greater than or equal to zero.

$$\forall x (x \geq 0)$$

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

$\exists x \dots$ there exists at least one thing x such that ...

$\forall x \dots$ for all x , it's the case that ...

$$\lim_{x \rightarrow a} f(x) = L$$

```
(forall (\epsilon) (if (> \epsilon 0)
  (exists (\delta) (and (> \delta 0)
    (forall (x) (if (< (dist x a) \delta)
      (< (dist (f x) L) \epsilon)))))))
```

$$\forall \epsilon (\epsilon > 0 \rightarrow \exists \delta (\delta > 0 \wedge \forall x (d(x, a) < \delta \rightarrow d(f(x), L) < \epsilon)))$$

Every natural number is greater than or equal to zero.

$$\forall x (x \geq 0)$$

Every positive integer x is less-than-or-equal-to a positive integer y .

But Now the Deeper Challenge:

Add Two Quantifiers to the Pure Predicate Calculus,

Which Yields \mathcal{L}_1 First-order Logic = Predicate Calculus *simpliciter*

$\exists x \dots$ there exists at least one thing x such that ...

$\forall x \dots$ for all x , it's the case that ...

$$\lim_{x \rightarrow a} f(x) = L$$

```
(forall (\epsilon) (if (> \epsilon 0)
  (exists (\delta) (and (> \delta 0)
    (forall (x) (if (< (dist x a) \delta)
      (< (dist (f x) L) \epsilon)))))))
```

$$\forall \epsilon (\epsilon > 0 \rightarrow \exists \delta (\delta > 0 \wedge \forall x (d(x, a) < \delta \rightarrow d(f(x), L) < \epsilon)))$$

Every natural number is greater than or equal to zero.

$$\forall x (x \geq 0)$$

Every positive integer x is less-than-or-equal-to a positive integer y .

$$\forall x \exists y (x \leq y) \quad \forall x \exists y (\leq (x, y))$$

The Shoulders Available to Gödel for Standing Upon

...

Completeness Theorem for The Propositional Calculus

Let Γ be a set $\{\phi_1, \phi_2, \dots\}$ of formulae in the the propositional calculus. Then either all of Γ are satisfiable, or the conjunction up to and including the point k (i.e. $\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k$) of failure is refutable.

Completeness Theorem for The Propositional Calculus

Let Γ be a set $\{\phi_1, \phi_2, \dots\}$ of formulae in the the propositional calculus. Then either all of Γ are satisfiable, or the conjunction up to and including the point k (i.e. $\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k$) of failure is refutable.

Completeness Theorem for The Propositional Calculus

Let Γ be a set $\{\phi_1, \phi_2, \dots\}$ of formulae in the the propositional calculus. Then either all of Γ are satisfiable, or the conjunction up to and including the point k (i.e. $\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k$) of failure is refutable.

Let Γ be a set $\{\phi_1, \phi_2, \dots\}$ of formulae in the the propositional calculus. Then either all of Γ can be simultaneously true in some scenario, or the conjunction up to and including the point k (i.e. $\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k$) of failure is **refutable** (i.e. $\vdash \neg(\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k)$).

What does the
Completeness Theorem
say?

...

Completeness Theorem as an Equation

In first-order logic: NECESSARY TRUTH = PROVABILITY.

Completeness Theorem, More Precisely Put

For every first-order statement ϕ : if ϕ is a necessary or absolute truth (i.e. true in any scenario whatsoever), then ϕ is provable.

And the version Gödel targeted,
and proved:

For every first-order statement ϕ : Either ϕ is true in some scenario, or ϕ is refutable (= it's negation $\neg\phi$ can be proved).

GCT

The Proof-Sketch

The Proof-Sketch

To prove the theorem in the case of first-order logic ($= \mathcal{L}_1$), we need to show that given any set Γ of formulae in first-order logic, either there's a scenario on which every member of this set is true; otherwise, there is a refutation of the set, i.e. a proof from the set to an outright contradiction $\phi \wedge \neg\phi$. We can accomplish this by finding a procedure \mathcal{P} that first takes the set in question and goes hunting for a scenario that does the trick. If the scenario is found, we're done. But, if such a scenario *can't* be found, then our procedure moves on to find a proof of a contradiction from Γ !

How?! The procedure \mathcal{P} is the building out of a truth tree! If all the branches in the tree close, then the finding of a proof of a contradiction uses **resolution**, and the **resolution guarantee**. The guarantee is that if you have a set of formulae that can't be true in any scenario, resolution applied to the set finds a contradiction $\perp = \zeta \wedge \neg\zeta = \{\}$. **QED**

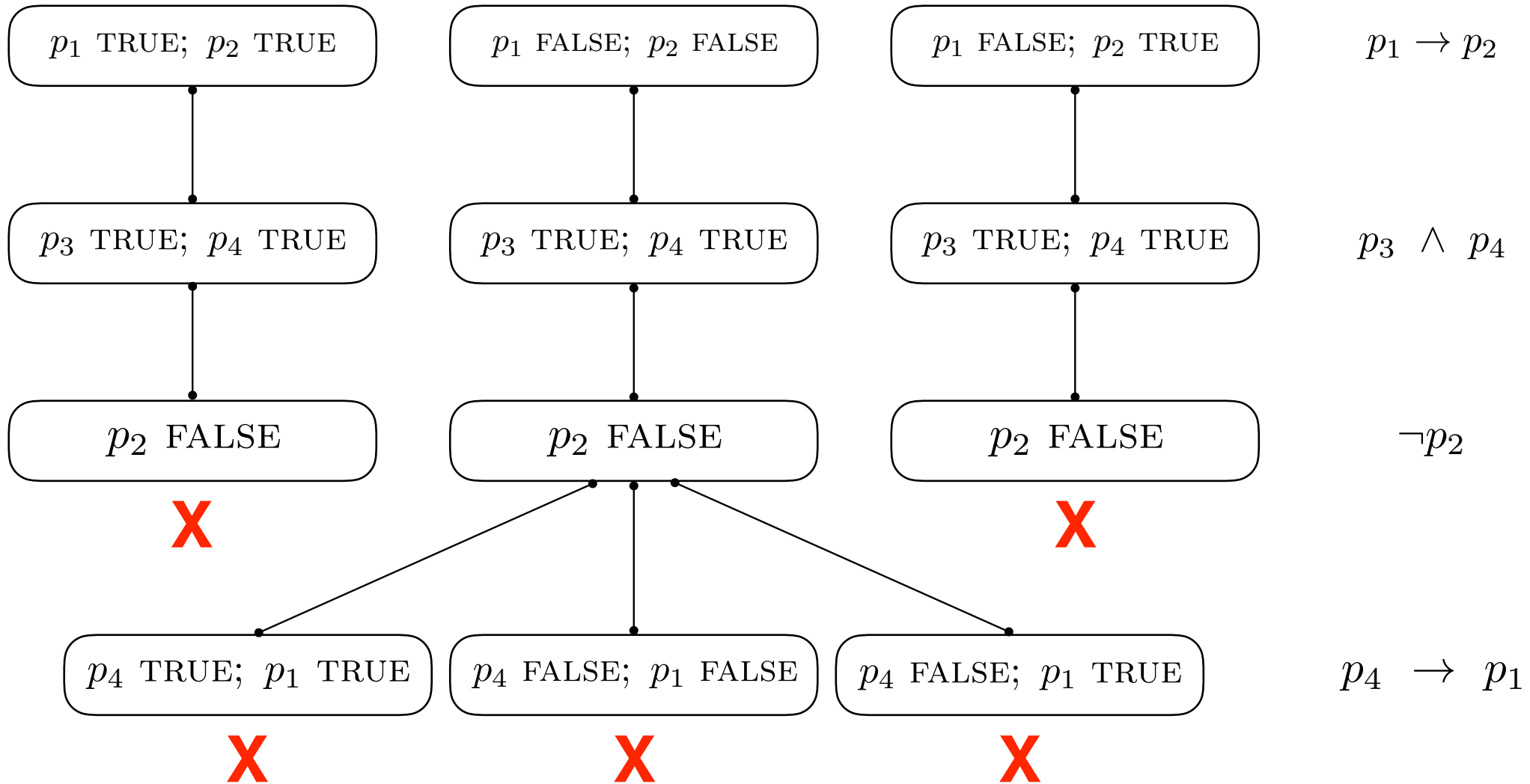
The Proof-Sketch

To prove the theorem in the case of first-order logic ($= \mathcal{L}_1$), we need to show that given any set Γ of formulae in first-order logic, either there's a scenario on which every member of this set is true; otherwise, there is a refutation of the set, i.e. a proof from the set to an outright contradiction $\phi \wedge \neg\phi$. We can accomplish this by finding a procedure \mathcal{P} that first takes the set in question and goes hunting for a scenario that does the trick. If the scenario is found, we're done. But, if such a scenario *can't* be found, then our procedure moves on to find a proof of a contradiction from Γ !

See LAMA-BDLAHGHS.

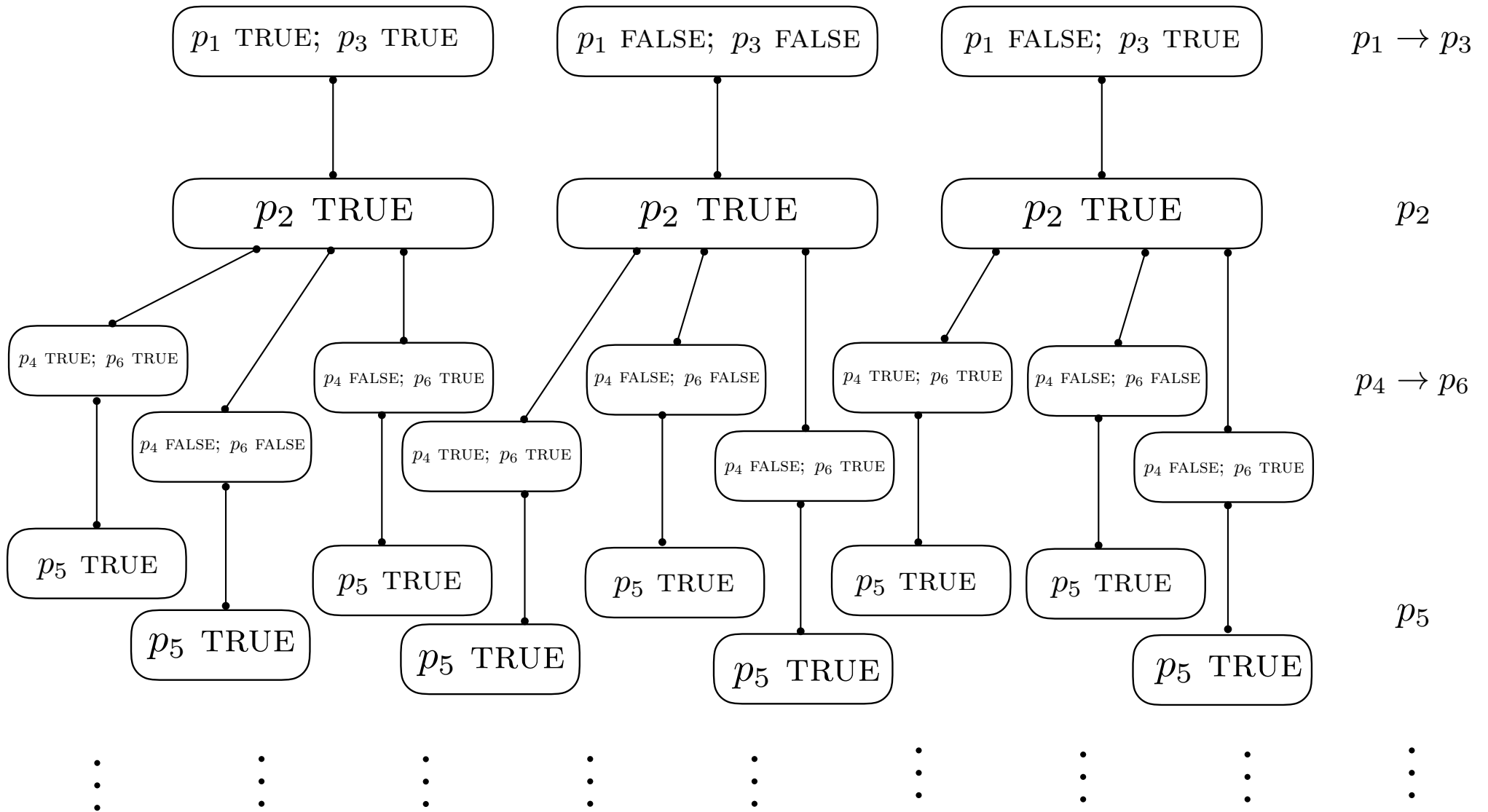
How?! The procedure \mathcal{P} is a search tree! If all the branches in the tree lead to a contradiction, then the set is refuted. The contradiction uses **resolution**, and the **resolution guarantee**. The guarantee is that if you have a set of formulae that can't be true in any scenario, resolution applied to the set finds a contradiction $\perp = \zeta \wedge \neg\zeta = \{\}$. **QED**

$$\Gamma := \{p_1 \rightarrow p_2, p_3 \wedge p_4, \neg p_2, p_4 \rightarrow p_1, \dots\}$$



Therefore, there is no scenario in which all of the formulae are true!

$$\Gamma := \{p_1 \rightarrow p_3, p_2, p_4 \rightarrow p_6, p_5, p_7 \rightarrow p_9, p_8, \dots\}$$



Therefore, since we can travel to infinity, there is a scenario in which all of the formulae are true: any infinite path down will do.

Ah, but can we travel to infinity? The assumption that there *is* an infinite branch here is based on König's Lemma ...

Toward König's Lemma as Train Travel

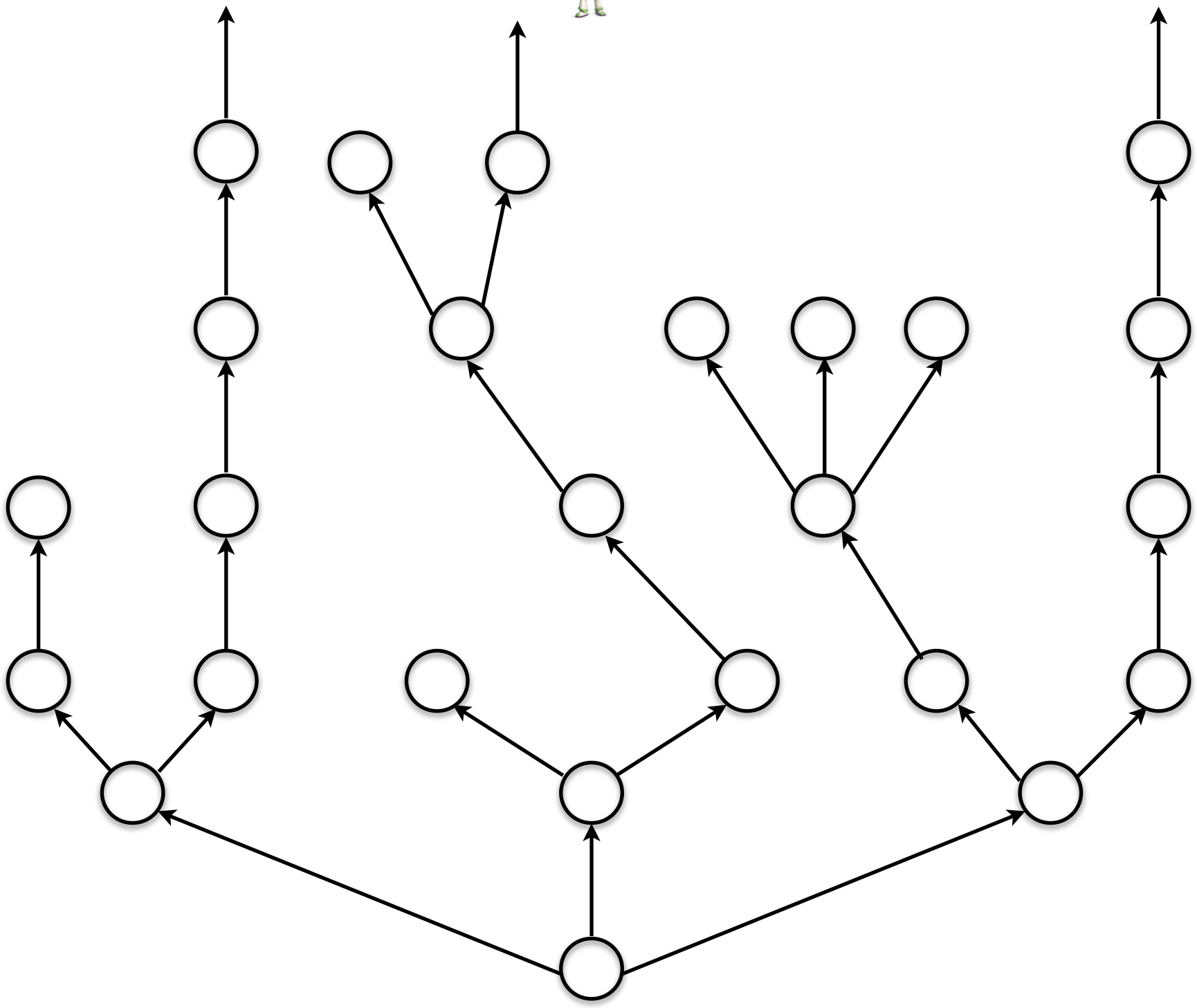


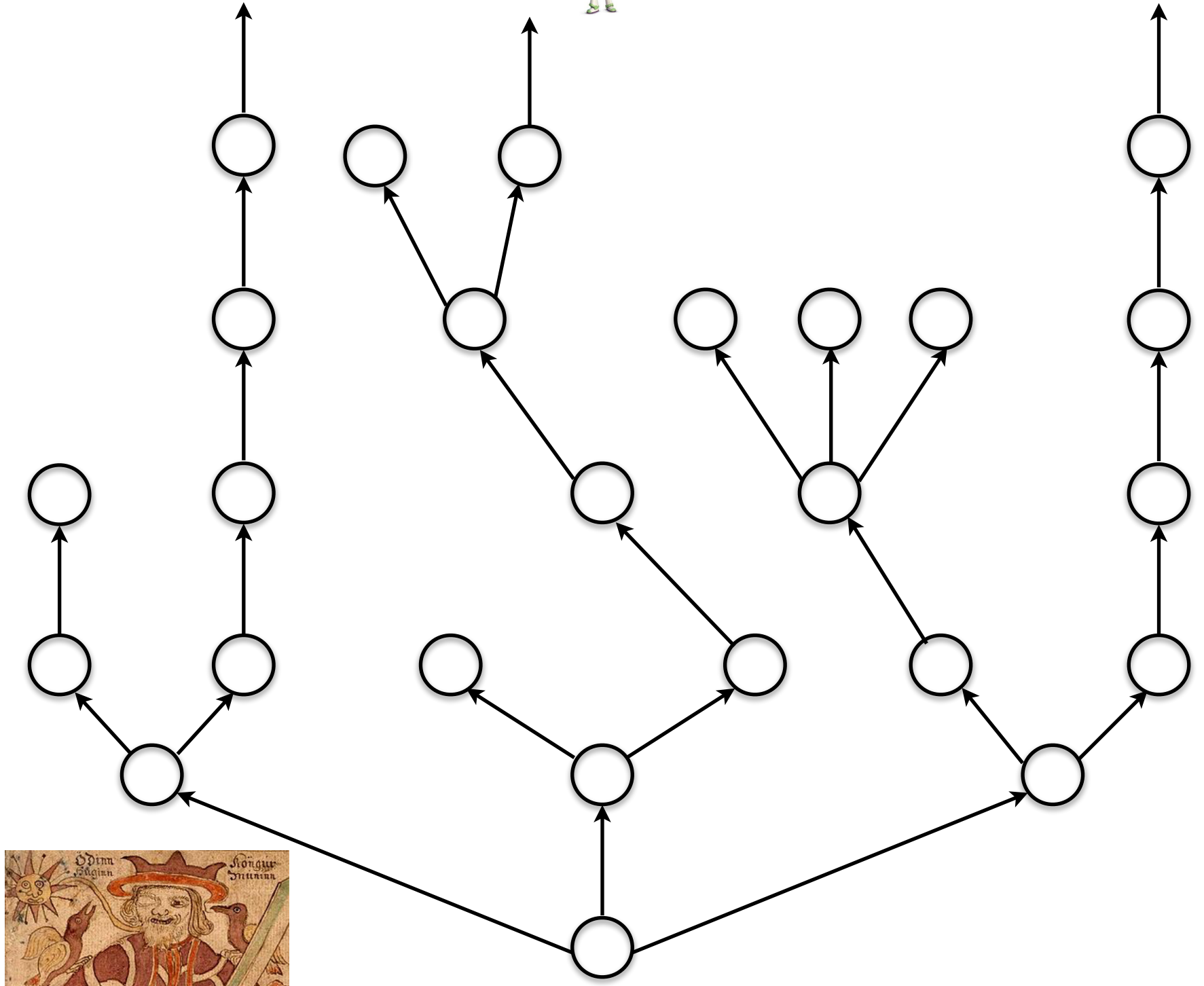
“To infinity and beyond!”

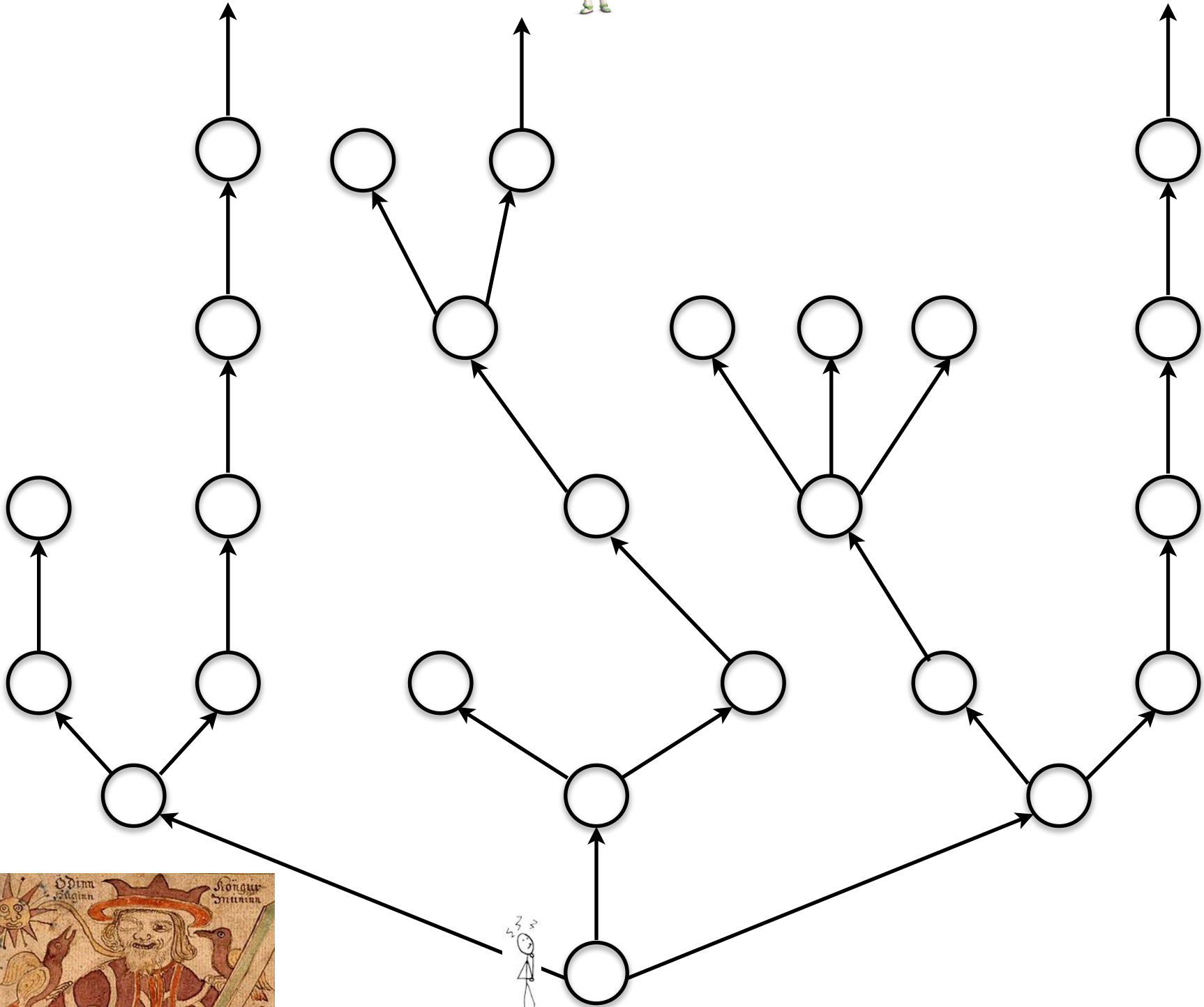


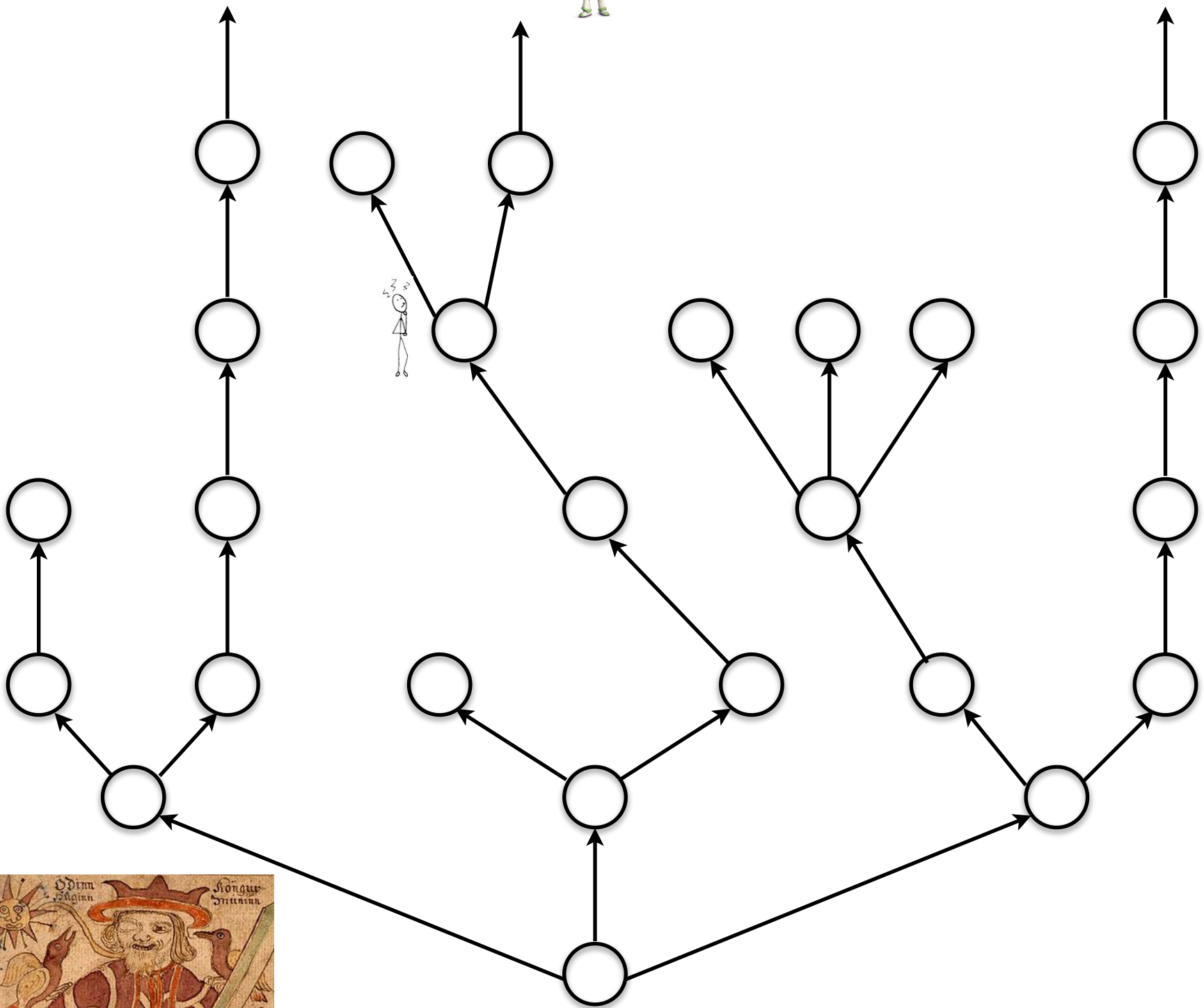
König's Lemma (train-travel version)

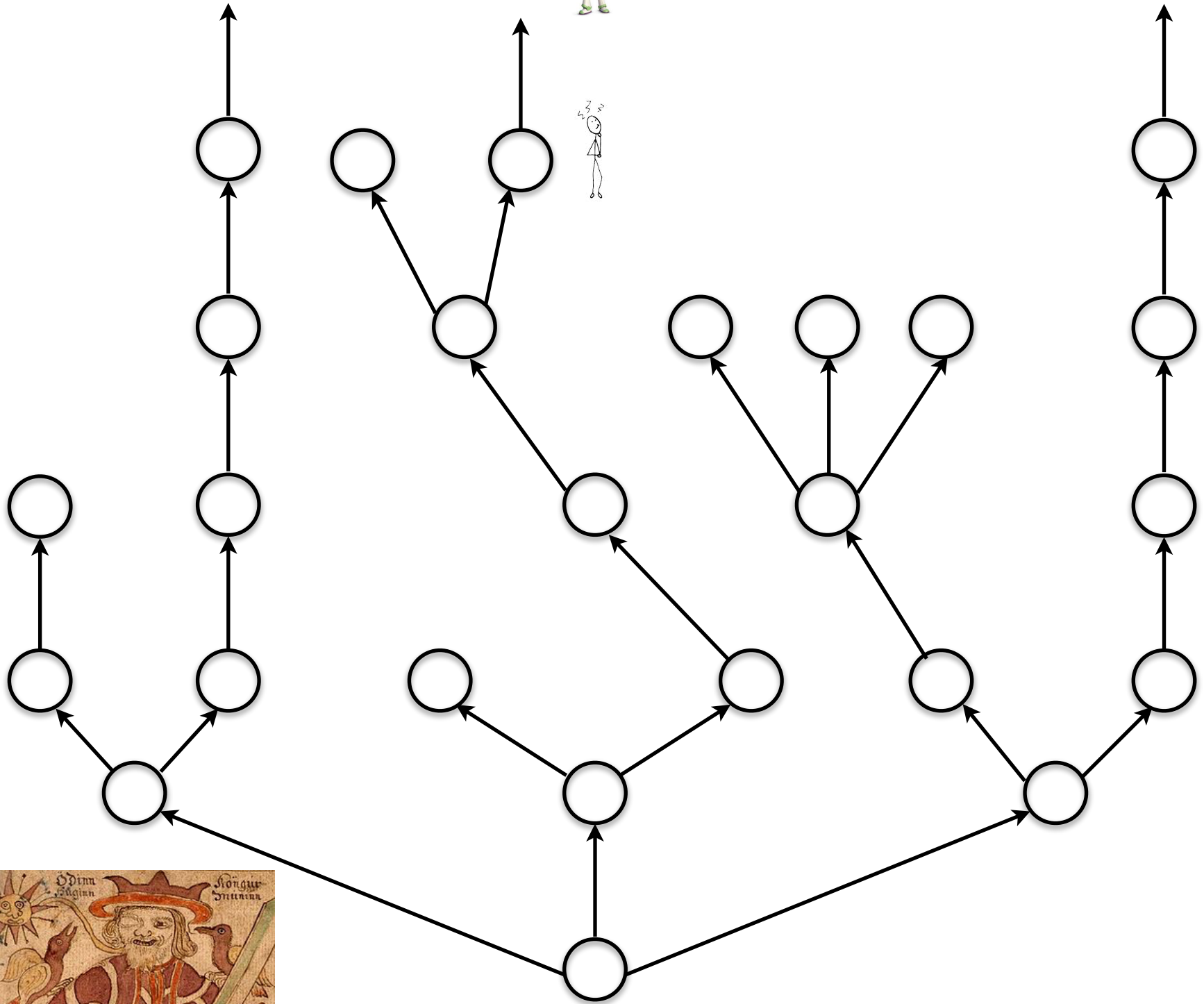
In a one-way train-travel map with finitely many options leading from each station, if there are partial paths forward of every finite length, there is an *infinite* path (= a path “to infinity”).

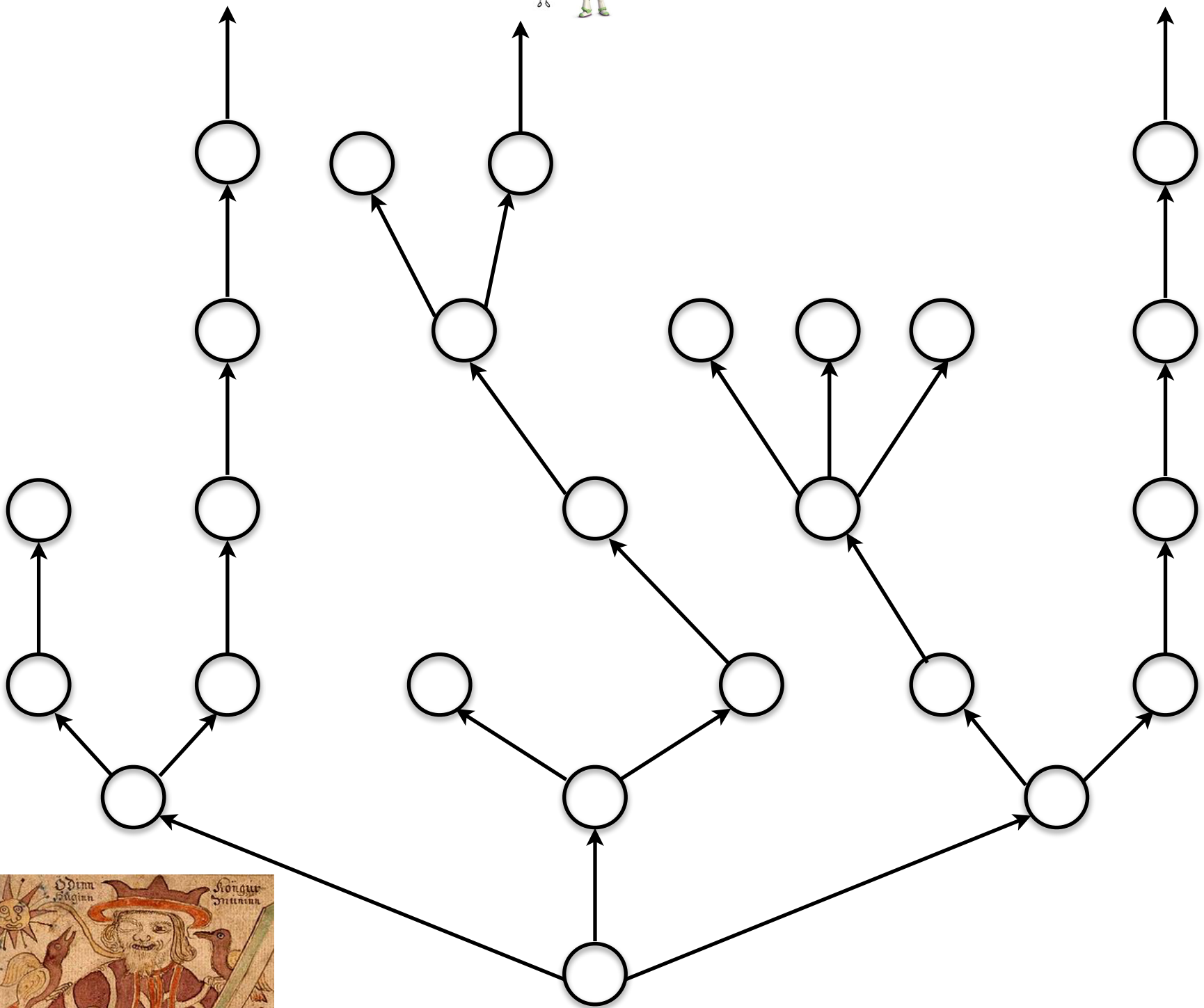












Exercise 2:

Is there an algorithm for traveling this way?

Exercise 2:

Is there an algorithm for traveling this way?

No. This strategy for travel is beyond the reach of constructive mathematics/standard computation.

Exercise 2:

Is there an algorithm for traveling this way?

No. This strategy for travel is beyond the reach of constructive mathematics/standard computation.

(Does it not then follow, assuming that humans can find and “use” a provably correct strategy for this travel, that humans can’t be fundamentally computing machines?)

NY-Centric Proof of the Lemma

(that there is an infinite branch)

Proof: We are seeking to prove that there is an infinite path (= that you can keep going forward forever = that the number of your stops forward are the size of \mathbf{Z}^+).

To begin, assume the antecedent of the theorem (i.e. that, (1), there are finitely many options leading from each station, and that, (2), in the map there are partial paths forward of every finite size).

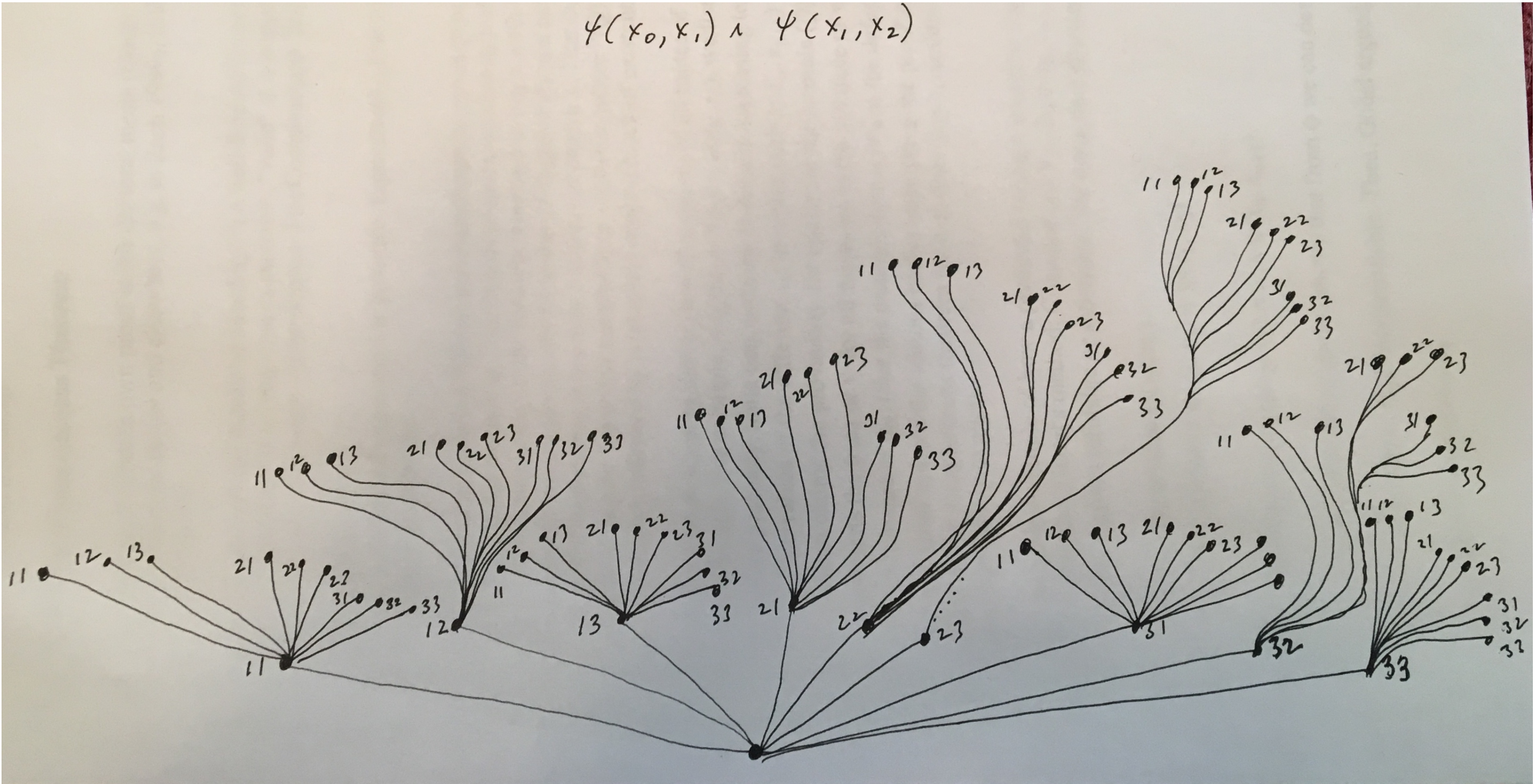
Now, you are standing at Penn Station (S_1), facing k options. At least one of these options must lead to partial paths of arbitrary size (the size of any m in \mathbf{Z}^+). (**Sub-Proof:** Suppose otherwise for indirect proof. Then there is some positive integer n that places a ceiling on the size of partial paths that can be reached. But this violates (2) — contradiction.) Proceed to choose one of these options that lead to partial paths of arbitrary size. You are now standing at a new station (S_2), one stop after Penn Station. At least one of these options must lead to partial parts of arbitrary size (the size of any m in \mathbf{Z}^+). (**Sub-Proof:** Suppose otherwise for indirect proof ...)

Since you can iterate this forever, you'll be on an infinite trip to infinity! Buzz will be happy.

QED

Simple Buzz-Lightyear-Like Branch

$$\psi(x_0, x_1) \wedge \psi(x_1, x_2)$$



Gödel as Giant: Some Evidence

THE DISCOVERY OF MY COMPLETENESS PROOFS

LEON HENKIN

Dedicated to my teacher, Alonzo Church, in his 91st year.

§1. Introduction. This paper deals with aspects of my doctoral dissertation¹ which contributed to the early development of model theory. What was of use to later workers was less the results of my thesis, than the method by which I proved the completeness of first-order logic—a result established by Kurt Gödel in *his* doctoral thesis 18 years before.²

The ideas that fed my discovery of this proof were mostly those I found in the teachings and writings of Alonzo Church. This may seem curious, as his work in logic, and his teaching, gave great emphasis to the constructive character of mathematical logic, while the model theory to which I contributed is filled with theorems about very large classes of mathematical structures, whose proofs often by-pass constructive methods.

Another curious thing about my discovery of a new proof of Gödel's completeness theorem, is that it arrived in the midst of my efforts to prove an entirely different result. Such "accidental" discoveries arise in many parts of scientific work. Perhaps there are regularities in the conditions under which such "accidents" occur which would interest some historians, so I shall try to describe in some detail the accident which befell me.

Received November 17, 1995, and in revised form, January 4, 1996.

Gödel as Giant: Some Evidence

THE DISCOVERY OF MY COMPLETENESS PROOFS

LEON HENKIN

Dedicated to my teacher, Alonzo Church, in his 91st year.

§1. Introduction. This paper deals with aspects of my doctoral dissertation¹ which contributed to the early development of model theory. What was of use to later workers was less the results of my thesis, than the method by which I proved the completeness of first-order logic—a result established by Kurt Gödel in *his* doctoral thesis 18 years before.²

The ideas that fed my discovery of this proof were mostly those I found in the teachings and writings of Alonzo Church. This may seem curious, as his work in logic, and his teaching, gave great emphasis to the constructive character of mathematical logic, while the model theory to which I contributed is filled with theorems about very large classes of mathematical structures, whose proofs often by-pass constructive methods.

Another curious thing about my discovery of a new proof of Gödel's completeness theorem, is that it arrived in the midst of my efforts to prove an entirely different result. Such "accidental" discoveries arise in many parts of scientific work. Perhaps there are regularities in the conditions under which such "accidents" occur which would interest some historians, so I shall try to describe in some detail the accident which befell me.

Received November 17, 1995, and in revised form, January 4, 1996.

But How'd He Handle All of \mathcal{L}_1 ?

But How'd He Handle All of \mathcal{L}_1 ?

Gödel proves the lemma that if the GCT holds for formula of degree j , GCT holds of formulae of degree $j+1$. So the challenge reduces to formulae of degree 1:

But How'd He Handle All of \mathcal{L}_1 ?

Gödel proves the lemma that if the GCT holds for formula of degree j , GCT holds of formulae of degree $j+1$. So the challenge reduces to formulae of degree 1:

$$\forall x_1, x_2, \dots, x_k \exists y_1, y_2, \dots, y_m \phi(x_1, x_2, \dots, x_k, y_1, y_2, \dots, y_m)$$

But How'd He Handle All of \mathcal{L}_1 ?

Gödel proves the lemma that if the GCT holds for formula of degree j , GCT holds of formulae of degree $j+1$. So the challenge reduces to formulae of degree 1:

$$\forall x_1, x_2, \dots, x_k \exists y_1, y_2, \dots, y_m \phi(x_1, x_2, \dots, x_k, y_1, y_2, \dots, y_m)$$

How? By ingenious tree-building, which starts by creating an enumeration of new constants $c = c_1, c_2, \dots$ that becomes our “universe of discourse”/“domain of quantification.” Note that from c we can algorithmically generate an enumeration of tuples $c^t = \langle c \rangle_1, \langle c \rangle_2, \dots$ of any finite size. (Those of size k will work for the x -variables, and those of size n will work for the y -variables.) And now we can build a BIG tree at the level of the pure predicate calculus, looking for either a scenario that makes our formula true by traveling with Buzz to infinity, or getting all branches closed, in which case we turn back to the resolution guarantee! Let's make sense of this by hand on paper ...

Gödel's Completeness Theorem, Degree-1 Formulae 3/8/20

f.f.

$$\psi := \forall x_1, x_2, \dots, x_k \exists y_1, y_2, \dots, y_m \phi(x_1, x_2, \dots, x_k; y_1, y_2, \dots, y_m)$$



$$\rightarrow G := c_1, c_2, c_3, \dots \omega$$

then tuples of the size of the x_i variables (i.e. size k) in this progression

$$\rightarrow G^{t-k} := \langle c \rangle_1^k, \langle c \rangle_2^k, \langle c \rangle_3^k, \dots \omega$$

And, tuples of the size of the y_i variables (i.e. size m) in this progression

$$\rightarrow G^{t-m} := \langle c \rangle_1^m, \langle c \rangle_2^m, \langle c \rangle_3^m, \dots \omega$$

And here's my enumeration

$$\begin{aligned} &\langle c \rangle_1^k; \langle c \rangle_1^m, \langle c \rangle_2^k; \langle c \rangle_2^m, \langle c \rangle_1^k; \langle c \rangle_3^k, \dots \omega \\ &\langle c \rangle_2^k; \langle c \rangle_1^m, \langle c \rangle_2^k; \langle c \rangle_2^m, \langle c \rangle_2^k; \langle c \rangle_3^k, \dots \omega \end{aligned}$$

⋮

In streamlined form:

11	12	13	14	15	...	ω
21	22	23	24	25	...	ω
31	32	33	34	35	...	ω
41	42	43	44	45	...	ω
⋮	⋮	⋮	⋮	⋮		
ω	ω	ω	ω	ω		

Do you see a way to travel to ∞ in a tree based on some enumeration of this infinite array?

Gödel's Completeness Theorem, Degree-1 Formulae 3/8/20

f.f.

$$\psi := \forall x_1, x_2, \dots, x_k \exists y_1, y_2, \dots, y_m \phi(x_1, x_2, \dots, x_k; y_1, y_2, \dots, y_m)$$



$$G := c_1, c_2, c_3, \dots \omega$$

then tuples of the size of the x_i variables (i.e. size k) in this progression

$$G^{t-k} := \langle c \rangle_1^k, \langle c \rangle_2^k, \langle c \rangle_3^k, \dots \omega$$

And, tuples of the size of the y_i variables (i.e. size m) in this progression

$$G^{t-m} := \langle c \rangle_1^m, \langle c \rangle_2^m, \langle c \rangle_3^m, \dots \omega$$

And here's my enumeration

$$\begin{aligned} &\langle c \rangle_1^k; \langle c \rangle_1^m, \langle c \rangle_2^k; \langle c \rangle_2^m, \langle c \rangle_1^k; \langle c \rangle_3^k, \dots \omega \\ &\langle c \rangle_2^k; \langle c \rangle_1^m, \langle c \rangle_2^k; \langle c \rangle_2^m, \langle c \rangle_2^k; \langle c \rangle_3^m, \dots \omega \end{aligned}$$

⋮

In streamlined form:











11	12	13	14	15	...	ω
21	22	23	24	25	...	ω
31	32	33	34	35	...	ω
41	42	43	44	45	...	ω
⋮	⋮	⋮	⋮	⋮		
ω	ω	ω	ω	ω		

Do you see a way to travel to ω in a tree based on some enumeration of this infinite array?

Making this Concrete Courtesy of HyperSlate®

The screenshot shows the HyperSlate web interface. At the top, there is a dark navigation bar with the HyperSlate logo and several icons: home, menu, add, cloud, lock, undo, redo, refresh, and a dropdown menu currently set to "Straight". On the right side of the bar, a status bar indicates "GodelFormula1 [FIRST-ORDER-LOGIC]: Saved with 38 symbols." The main workspace is a light gray area. In the center, a green box labeled "assume" is connected by a vertical line to a larger white box with a purple header. The header contains the text "SAME FORMULA FOR SAT TESTING" followed by the logical formula $\forall x: \exists y: L(x, y) \wedge \neg L(x, y)$. Below the header, the text "from {SAME FORMULA FOR SAT TESTING}" is displayed.

Making this Concrete Courtesy of HyperSlate[®]

HyperSlate[®]          Straight  GodelFormula1 [FIRST-ORDER-LOGIC]: Saved with 38 symbols.

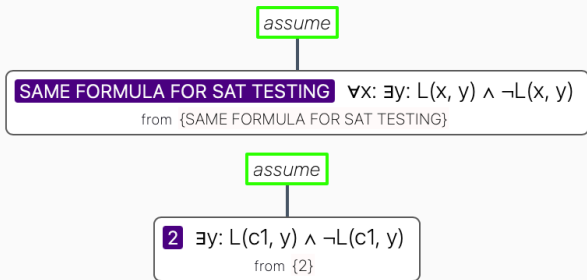








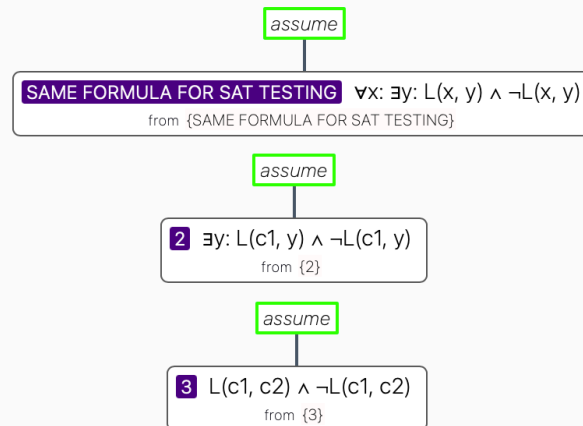


Diagram illustrating a logical derivation:

- Top box: **assume**
- Middle box: **SAME FORMULA FOR SAT TESTING** $\forall x: \exists y: L(x, y) \wedge \neg L(x, y)$
from {SAME FORMULA FOR SAT TESTING}
- Bottom box: **2** $\exists y: L(c1, y) \wedge \neg L(c1, y)$
from {2}

Making this Concrete Courtesy of HyperSlate[®]

HyperSlate[®]          Straight  GodelFormula1 [FIRST-ORDER-LOGIC]: Saved with 38 symbols.



Making this Concrete Courtesy of HyperSlate[®]

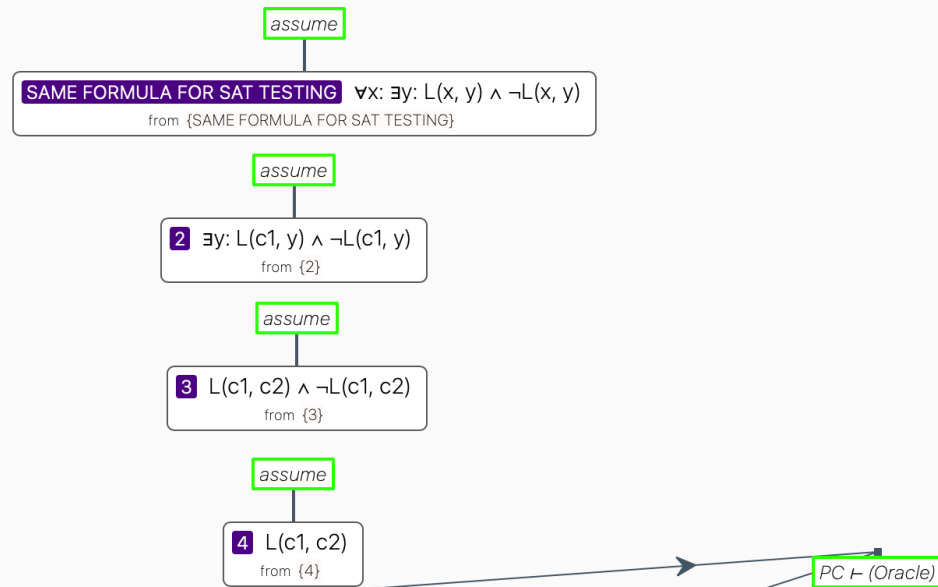
HyperSlate[®]



Straight



GodelFormula1 [FIRST-ORDER-LOGIC]: Saved with 38 symbols.



Making this Concrete Courtesy of HyperSlate®

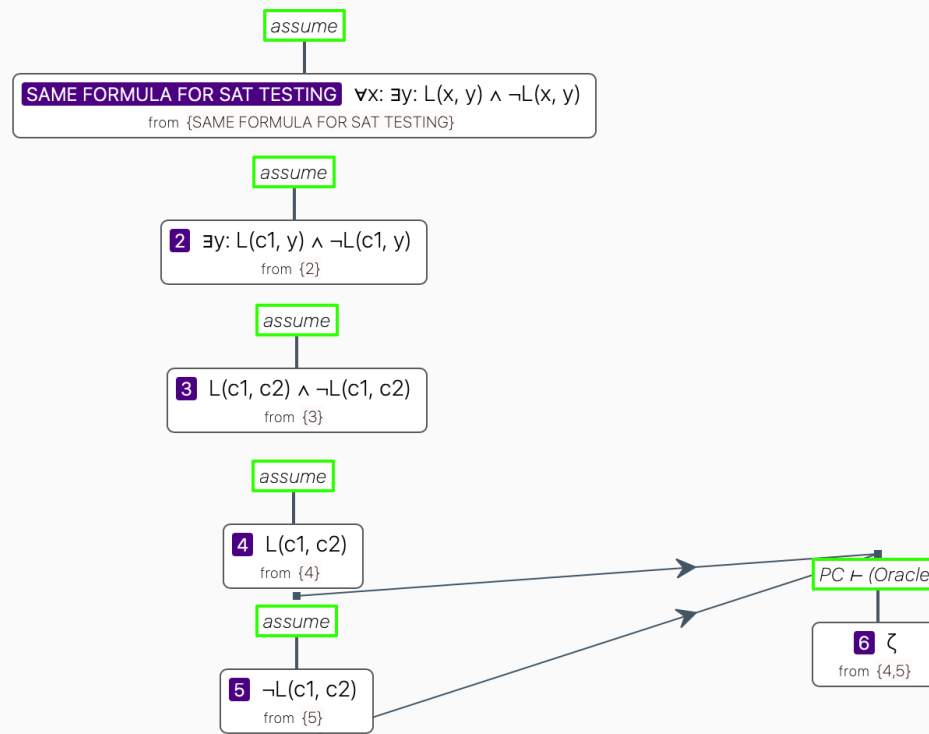
HyperSlate®



Straight



GodelFormula1 [FIRST-ORDER-LOGIC]: Saved with 38 symbols.



Making this Concrete Courtesy of HyperSlate[®]

The screenshot shows a Safari browser window displaying the HyperSlate web application. The address bar shows the URL `rpi.logicamodernapproach.com`. The browser tabs include "Sign in to hypergrader", "Editing GodelFormula1", and "teller 2ch8.pdf truth trees - Google Search". The HyperSlate interface features a toolbar with icons for home, list, add, save, undo, redo, and Bezier. A status bar at the top right indicates "GodelFormula1 [FIRST-ORDER-LOGIC]: Saved with 38 symbols." The main content area displays a logical proof diagram with the following steps:

- Root node: **assume** (green box)
- Node 1: **SAME FORMULA FOR SAT TESTING** $\forall x: \exists y: L(x, y) \wedge \neg L(x, y)$ from {SAME FORMULA FOR SAT TESTING}
- Node 2: **assume** (green box)
- Node 2: **2** $\exists y: L(c1, y) \wedge \neg L(c1, y)$ from {2}
- Node 3: **assume** (green box)
- Node 3: **3** $L(c1, c2) \wedge \neg L(c1, c2)$ from {3}
- Node 4: **assume** (green box)
- Node 4: **4** $L(c1, c2)$ from {4}
- Node 5: **assume** (green box)
- Node 5: **5** $\neg L(c1, c2)$ from {5}
- Node 6: **PC \vdash (Oracle)** (green box)
- Node 6: **6** ζ from {4,5}

Arrows indicate dependencies: Node 4 and Node 5 both point to Node 6. The diagram is displayed on a desktop environment with a dock at the bottom containing various application icons.

Making this Concrete Courtesy of HyperSlate[®]

The screenshot shows a Safari browser window displaying the HyperSlate web application. The address bar shows the URL `rpi.logicamodernapproach.com`. The browser tabs include "Sign in to hypergrader", "Editing GodelFormula1", and "teller 2ch8.pdf truth trees - Google Search". The HyperSlate interface features a toolbar with icons for home, list, add, save, undo, redo, and Bezier. A status bar at the top right indicates "GodelFormula1 [FIRST-ORDER-LOGIC]: Saved with 38 symbols." The main content area displays a logical proof diagram with the following steps:

- Step 1: `assume` (highlighted in green)
- Step 2: `SAME FORMULA FOR SAT TESTING` $\forall x: \exists y: L(x, y) \wedge \neg L(x, y)$ from {SAME FORMULA FOR SAT TESTING}
- Step 3: `assume` (highlighted in green)
- Step 4: `2` $\exists y: L(c1, y) \wedge \neg L(c1, y)$ from {2}
- Step 5: `assume` (highlighted in green)
- Step 6: `3` $L(c1, c2) \wedge \neg L(c1, c2)$ from {3}
- Step 7: `assume` (highlighted in green)
- Step 8: `4` $L(c1, c2)$ from {4}
- Step 9: `assume` (highlighted in green)
- Step 10: `5` $\neg L(c1, c2)$ from {5}
- Step 11: `PC ⊢ (Oracle)` (highlighted in green)
- Step 12: `6` ζ from {4,5}

The diagram shows arrows pointing from steps 8 and 10 to step 11, and from step 11 to step 12. The left sidebar contains a list of topics and a meeting link.

Making this Concrete Courtesy of HyperSlate[®]

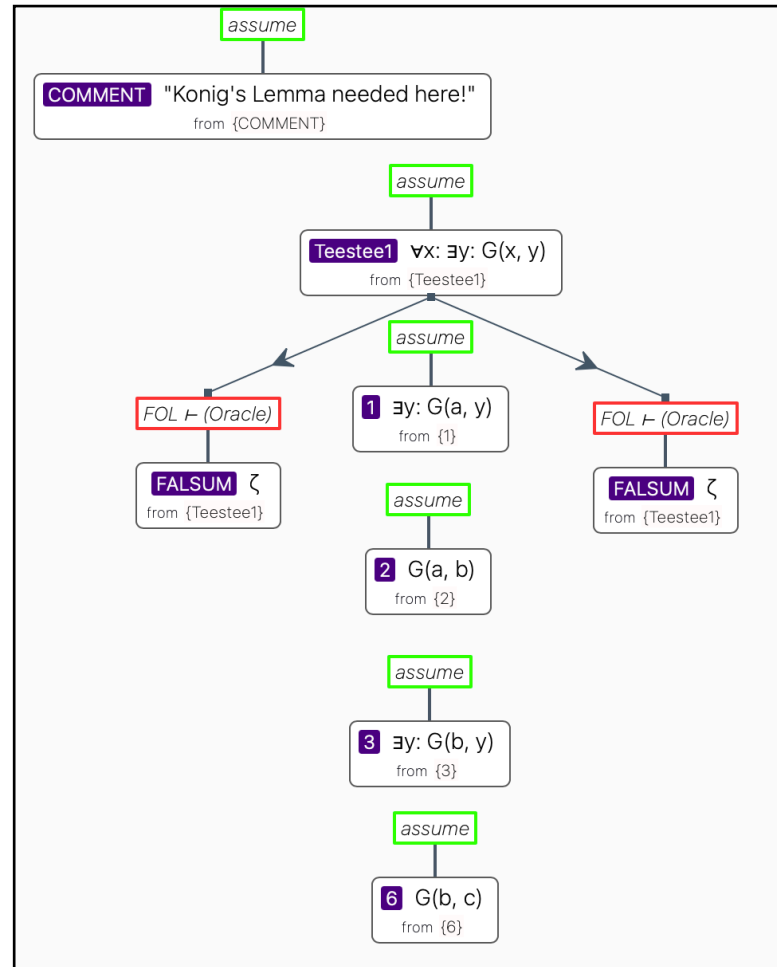
Volunteers to do the following Buzz-Lightyear-relevant formula, from earlier in the present slide deck?

$$\forall x \exists y y > x$$

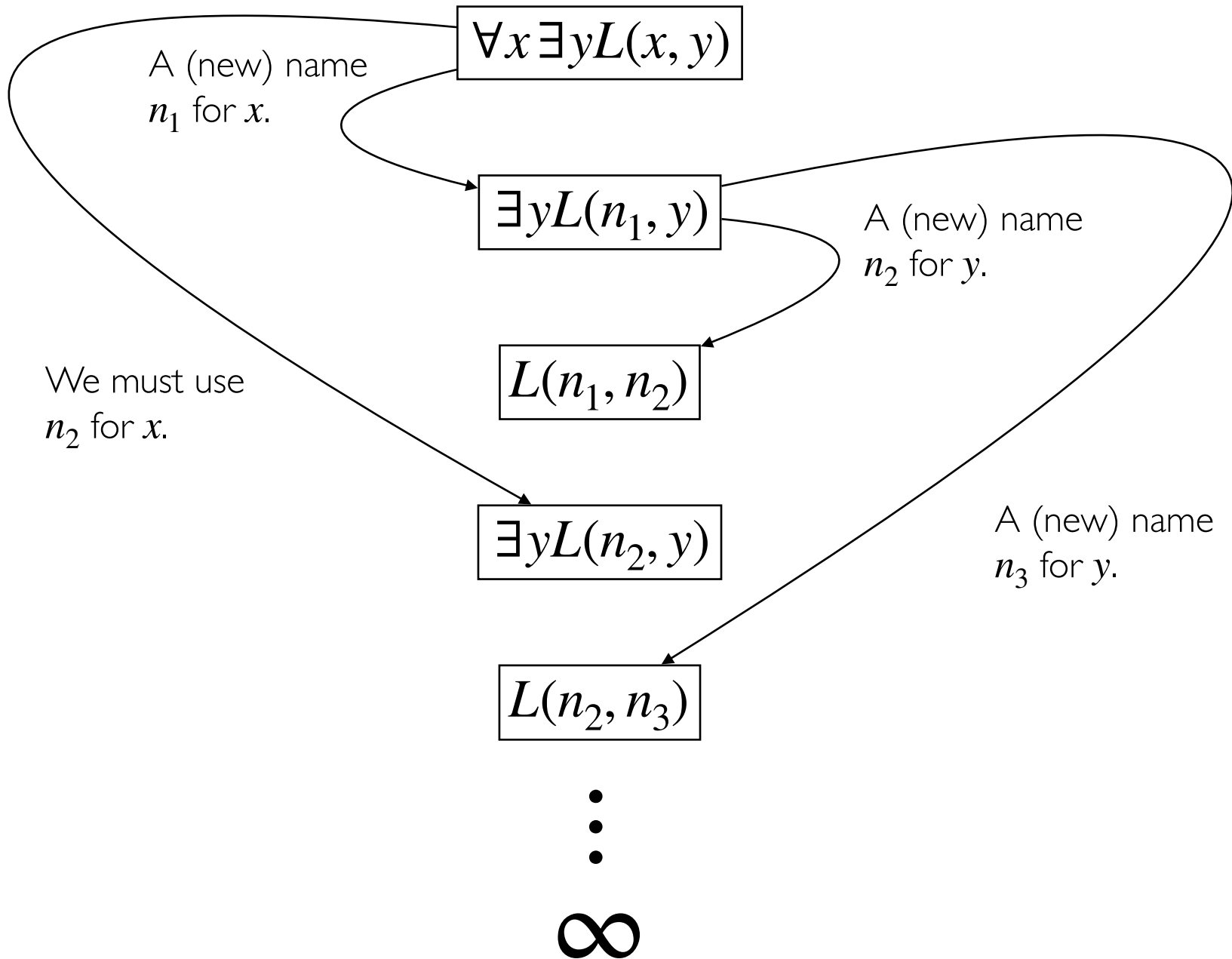
Making this Concrete

Courtesy of HyperSlate[®]

E.g.



•
•
•



For every x , there's a y s.t. x is less than y .

A (new) name
 n_1 for x .

There's a y s.t. object n_1 is less than y .

We must use
 n_2 for x .

Object n_1 is less than object n_2 .

A (new) name
 n_2 for y .

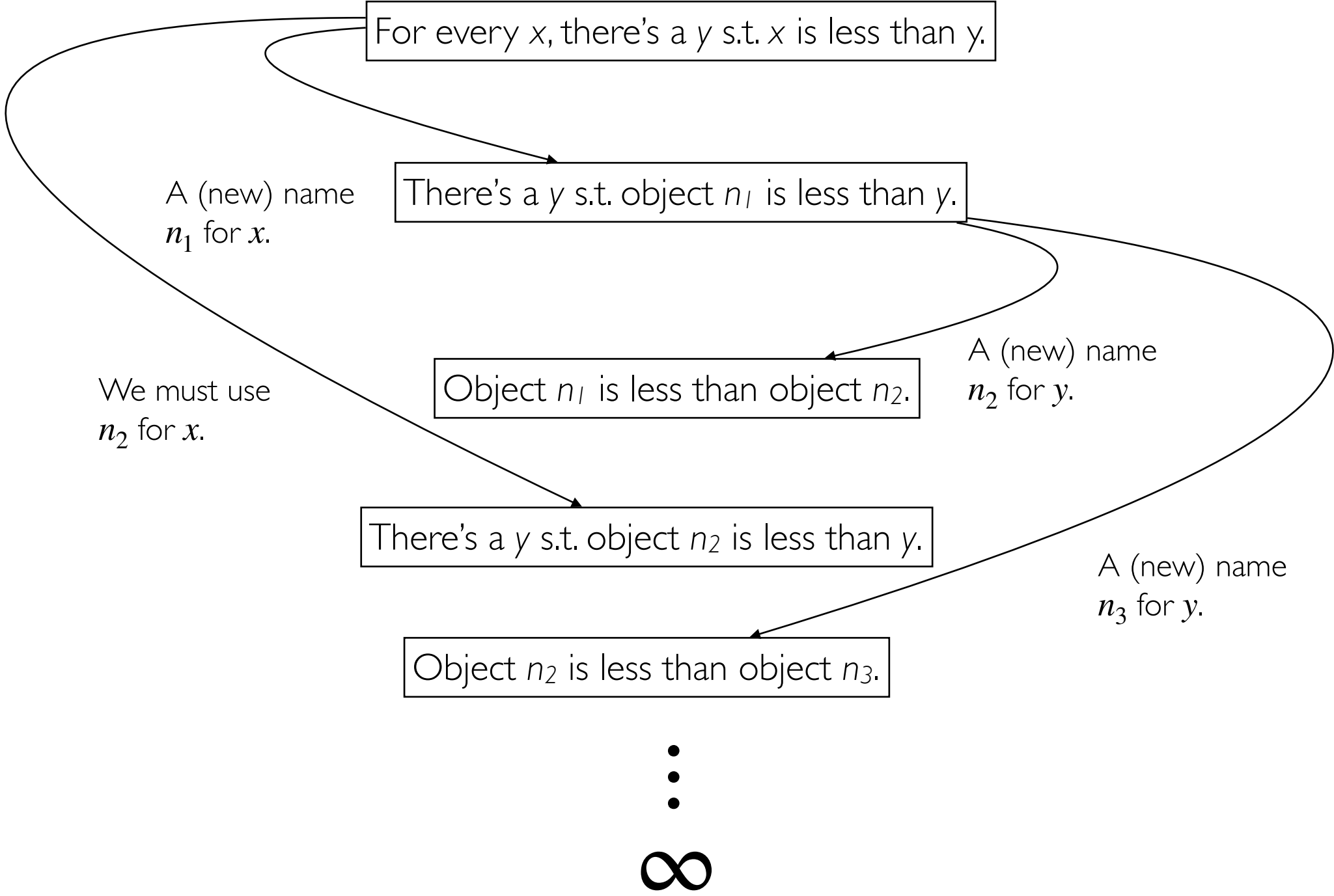
There's a y s.t. object n_2 is less than y .

A (new) name
 n_3 for y .

Object n_2 is less than object n_3 .

⋮

∞



slutten