

Introducing *Pure General Logic Programming (PGLP)*, in **HyperSlate[®]:HyperLog[®]**; **Review of All Inference Rules/ Schemata in PropCalc = \mathcal{L}_{PC}**

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

IFLAI
2/5/2024



Logic-and-AI in the news

...

Why Is Big Tech Still Cutting Jobs?

Profits are up and the economy is strong. But the tech industry faces two challenges — dealing with a frenetic work force expansion in the pandemic and building A.I.



Microsoft made job cuts in its video game division in January. Ruth Fremson/The New York Times

Mark Zuckerberg, the chief executive of Meta, said in a call with analysts last week that his company had to lay off employees and control costs “so we can invest in these long-term, ambitious visions around A.I.” He added that he had come to realize that “we operate better as a leaner company.”

From the end of 2019 to 2023, tech companies scrambled to keep up with an explosion of consumer demand, as people stuck at home splurged on new computers and spent much more time online. Apple, Amazon, Meta, Microsoft and Alphabet, Google’s parent company, added a total of more than 900,000 jobs.

When that boom ended, they were forced to adjust. Meta, Amazon, Microsoft, Google and Apple cut about 112,000 jobs from their respective peaks in 2021 and 2022. But they were still much bigger and more profitable than before the pandemic began.

Today, the five companies employ 2.16 million people, 71 percent more than they had before the pandemic. Combined, they generated \$1.63 trillion in sales in their most recent fiscal years, about 81 percent more revenue than five years earlier.

Why Is Big Tech Still Cutting Jobs?

Profits are up and the economy is strong. But the tech industry faces two challenges — dealing with a frenetic work force expansion in the pandemic and building A.I.



Microsoft made job cuts in its video game division in January. Ruth Fremson/The New York Times

Mark Zuckerberg, the chief executive of Meta, said in a call with analysts last week that his company had to lay off employees and control costs “so we can invest in these long-term, ambitious visions around A.I.” He added that he had come to realize that “we operate better as a leaner company.”

From the end of 2019 to 2023, tech companies scrambled to keep up with an explosion of consumer demand, as people stuck at home splurged on new computers and spent much more time online. Apple, Amazon, Meta, Microsoft and Alphabet, Google’s parent company, added a total of more than 900,000 jobs.

When that boom ended, they were forced to adjust. Meta, Amazon, Microsoft, Google and Apple cut about 112,000 jobs from their respective peaks in 2021 and 2022. But they were still much bigger and more profitable than before the pandemic began.

Today, the five companies employ 2.16 million people, 71 percent more than they had before the pandemic. Combined, they generated \$1.63 trillion in sales in their most recent fiscal years, about 81 percent more revenue than five years earlier.

RESEARCH

AlphaGeometry: An Olympiad-level AI system for geometry

17 JANUARY 2024

Trieu Trinh and Thang Luong

[🔗 Share](#)



Our AI system surpasses the state-of-the-art approach for geometry problems, advancing AI reasoning in mathematics

Logistics again ...

The Starting Code to Purchase in Bookstore

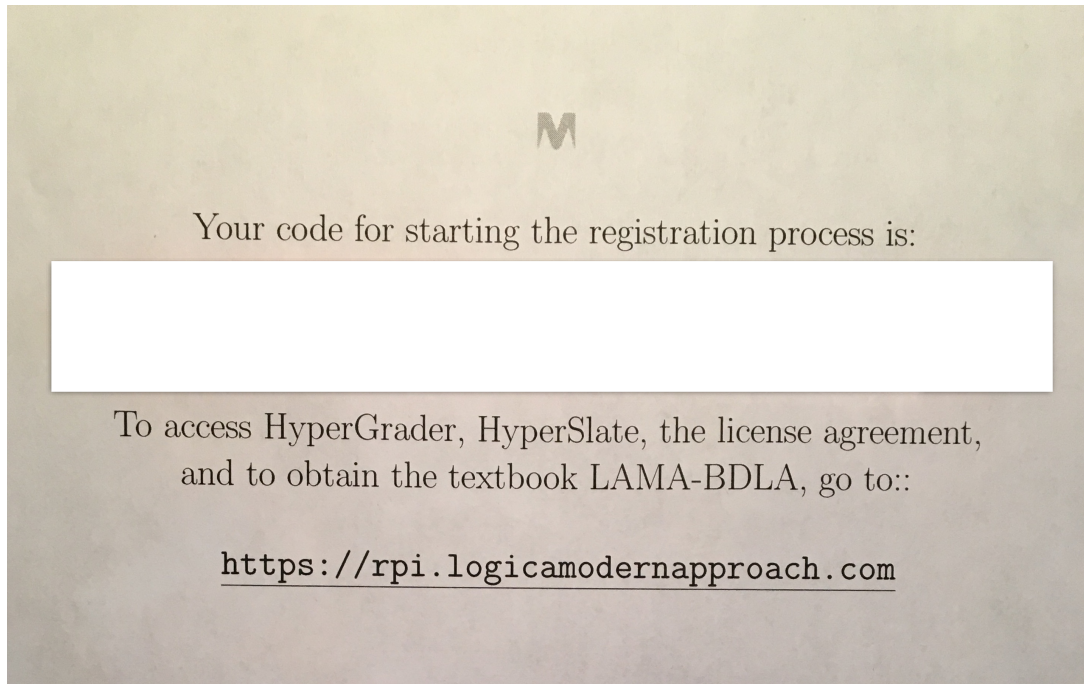
M

Your code for starting the registration process is:

To access HyperGrader, HyperSlate, the license agreement,
and to obtain the textbook LAMA-BDLA, go to::

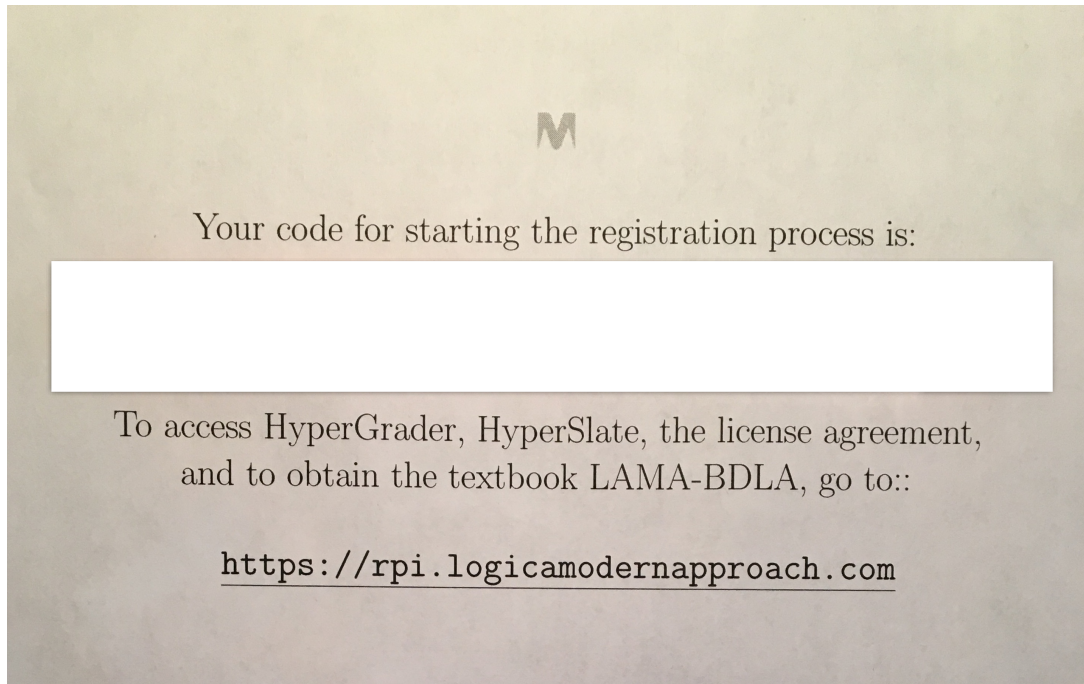
<https://rpi.logicamodernapproach.com>

The Starting Code to Purchase in Bookstore



Once seal broken on envelope, no return. Remember from first class, any reservations, opt for “Stanford” paradigm, with its software instead of LAMA™ paradigm!

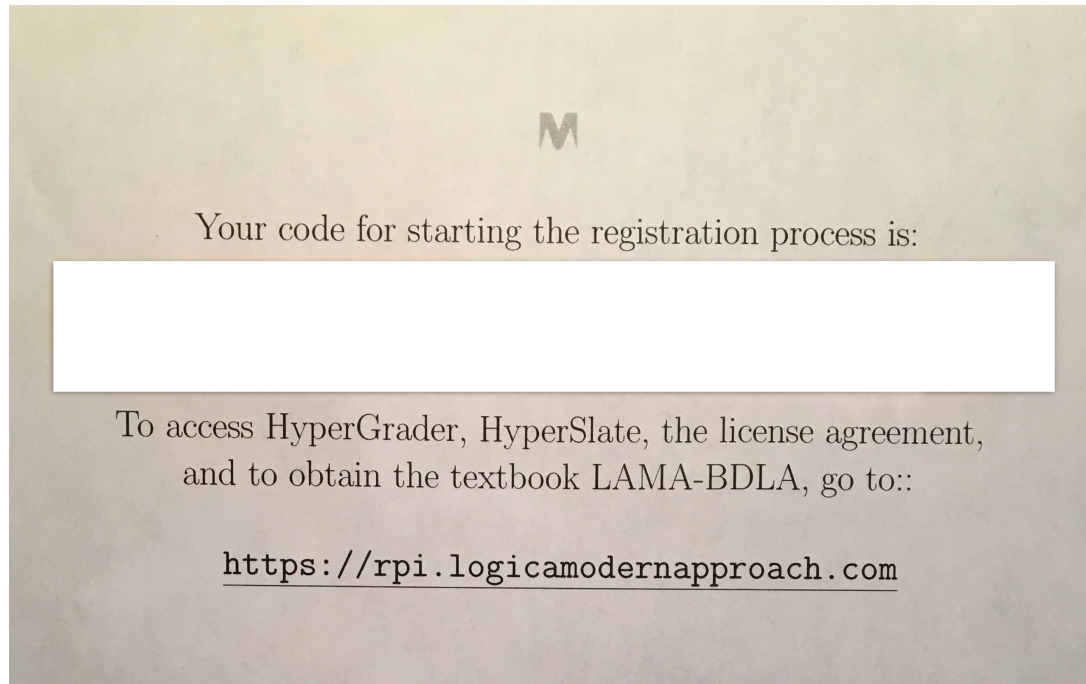
The Starting Code to Purchase in Bookstore



Once seal broken on envelope, no return. Remember from first class, any reservations, opt for “Stanford” paradigm, with its software instead of LAMA™ paradigm!

The email address you enter is case-sensitive!

The Starting Code to Purchase in Bookstore

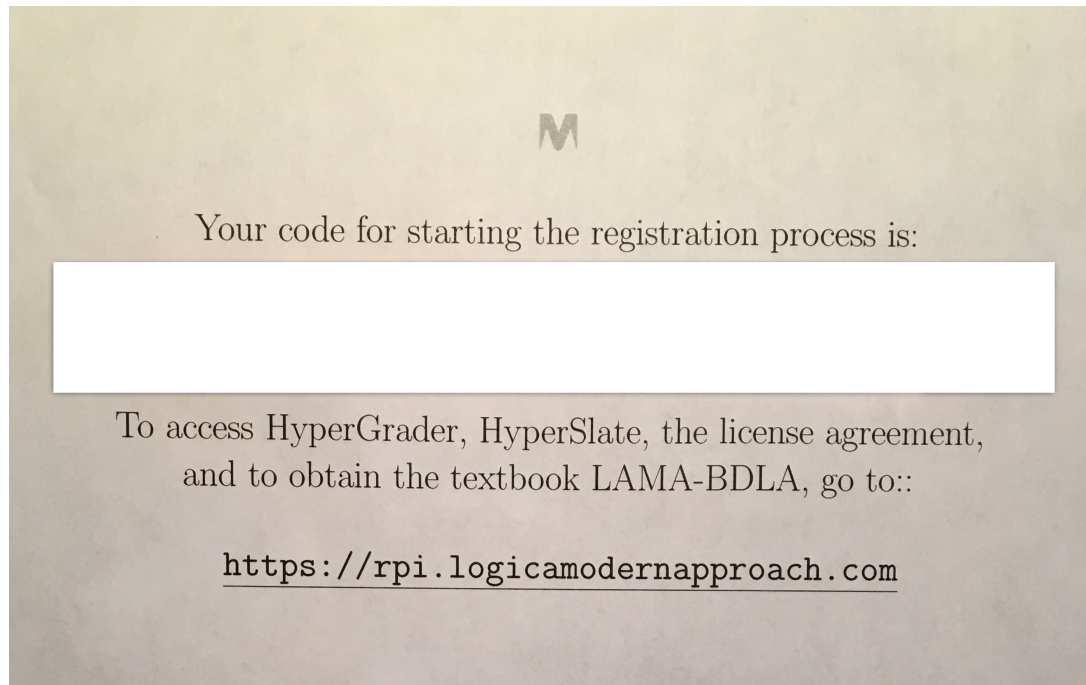


Once seal broken on envelope, no return. Remember from first class, any reservations, opt for “Stanford” paradigm, with its software instead of LAMA™ paradigm!

The email address you enter is case-sensitive!

Your OS and browser must be fully up-to-date; Chrome is the best choice, browser-wise (though I use Safari).

The Starting Code to Purchase in Bookstore



Once seal broken on envelope, no return. Remember from first class, any reservations, opt for “Stanford” paradigm, with its software instead of LAMA™ paradigm!

The email address you enter is case-sensitive!

Your OS and browser must be fully up-to-date; Chrome is the best choice, browser-wise (though I use Safari).

Watch that the link doesn't end up being classified as spam.

Introducing *Pure General Logic Programming (PGLP)*, in **HyperSlate[®]:HyperLog[®]**; **Review of All Inference Rules/ Schemata in PropCalc = \mathcal{L}_{PC}**

Selmer Bringsjord

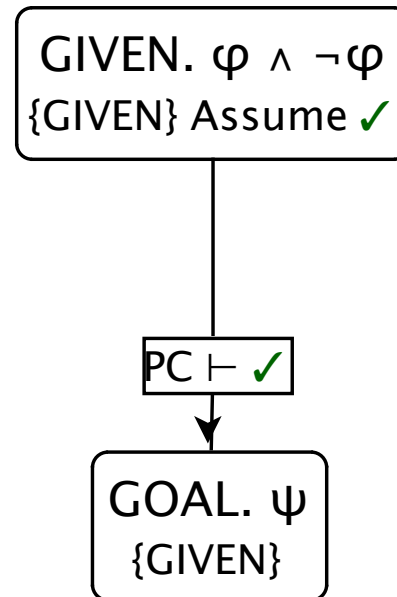
Rensselaer AI & Reasoning (RAIR) Lab
Department of Cognitive Science
Department of Computer Science
Lally School of Management & Technology
Rensselaer Polytechnic Institute (RPI)
Troy, New York 12180 USA

IFLAI
2/5/2024



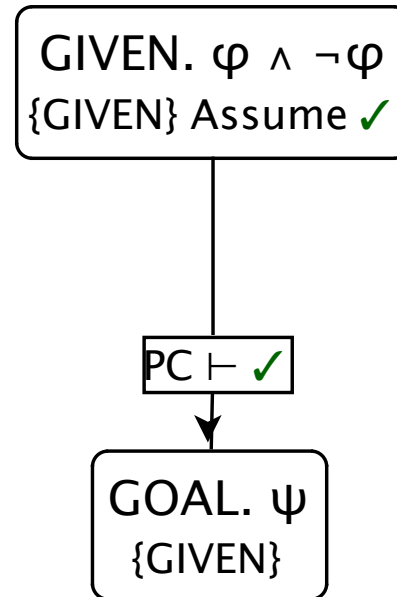
A computer program and its execution in
PGLP (@ level of propositional calculus):

A computer program and its execution in PGLP (@ level of propositional calculus):



A computer program and its execution in PGLP (@ level of propositional calculus):

Program:



A computer program and its execution in PGLP (@ level of propositional calculus):

Program:

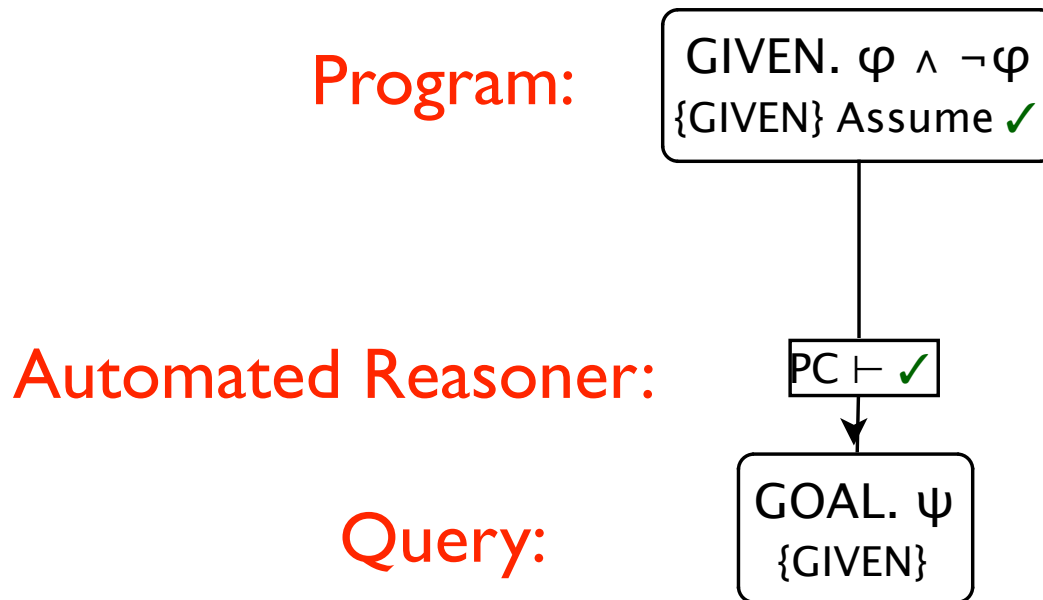
GIVEN. $\varphi \wedge \neg\varphi$
{GIVEN} Assume ✓

PC ⊢ ✓

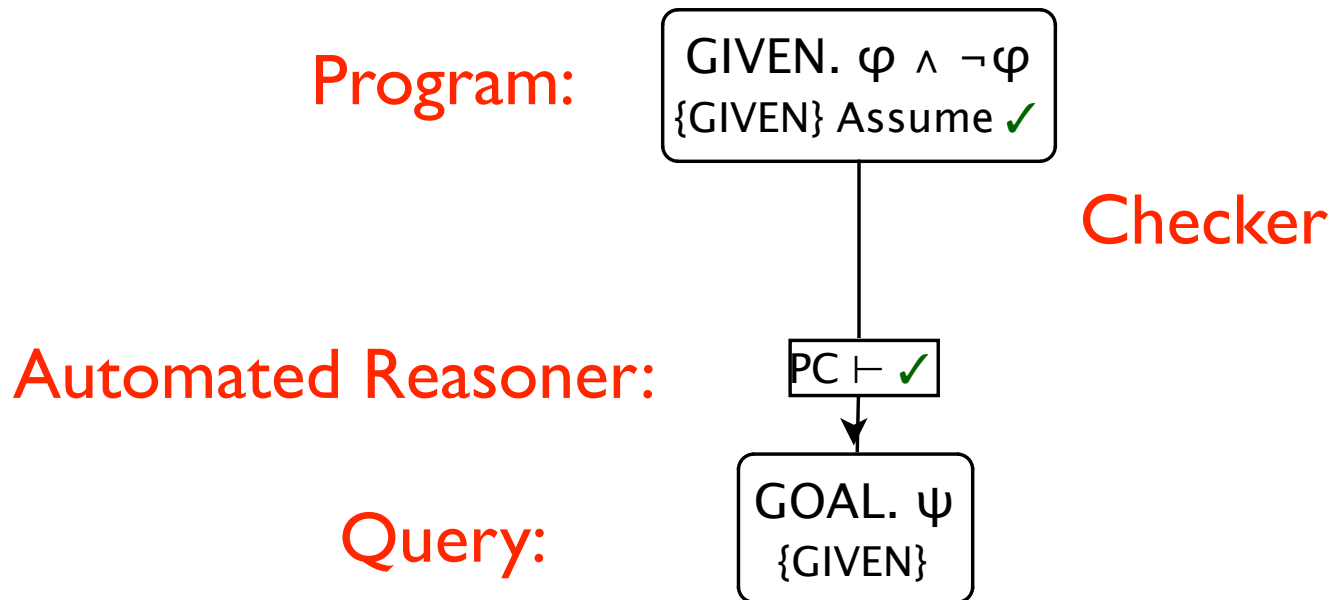
Query:

GOAL. ψ
{GIVEN}

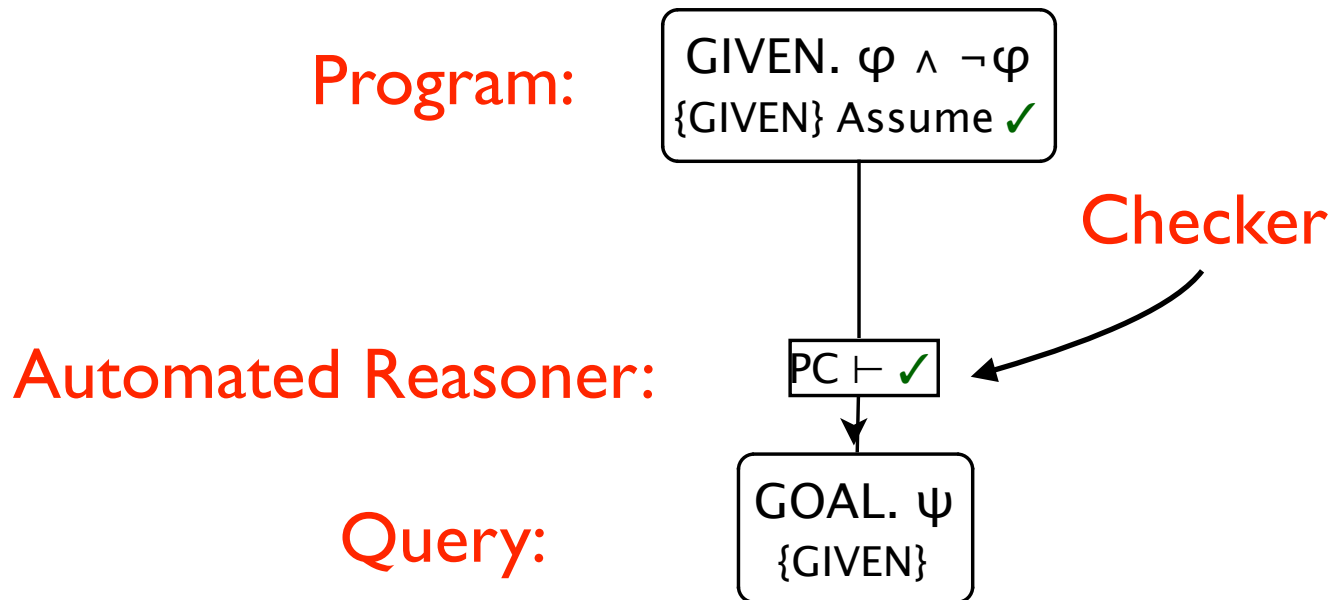
A computer program and its execution in PGLP (@ level of propositional calculus):



A computer program and its execution in PGLP (@ level of propositional calculus):



A computer program and its execution in PGLP (@ level of propositional calculus):



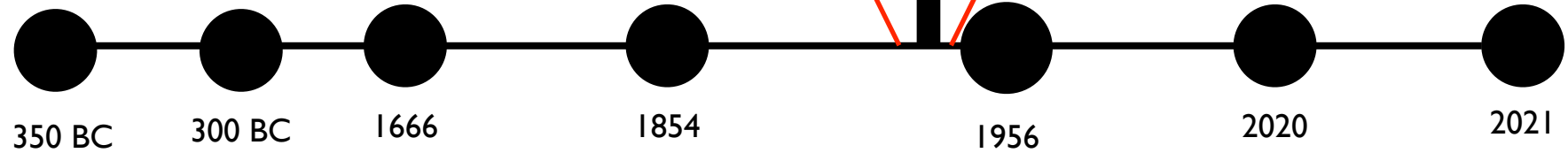
?

Entscheidungsproblem

“Universal Computational Logic”



Logic Theorist
(birth of modern logicist AI)



Euclid

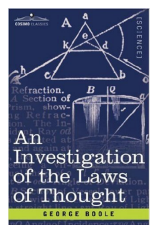


Organon



Leibniz

\int



Simon

Intro to (Formal) Logic (& AI)

T
h
e
S
i
n
g
u
l
a
r
i
t
y
?

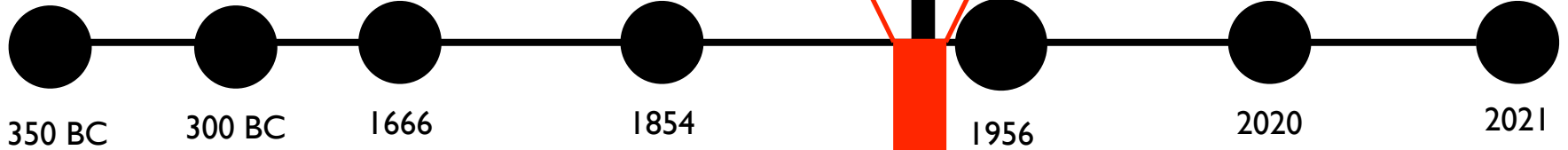
Entscheidungsproblem



“Universal Computational Logic”



Logic Theorist
(birth of modern logicist AI)



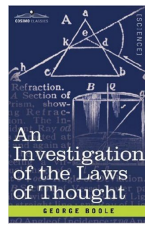
Euclid



Organon



Leibniz



Simon

Intro to (Formal) Logic (& AI)

T
h
e
S
i
n
g
u
l
a
r
i
t
y
?

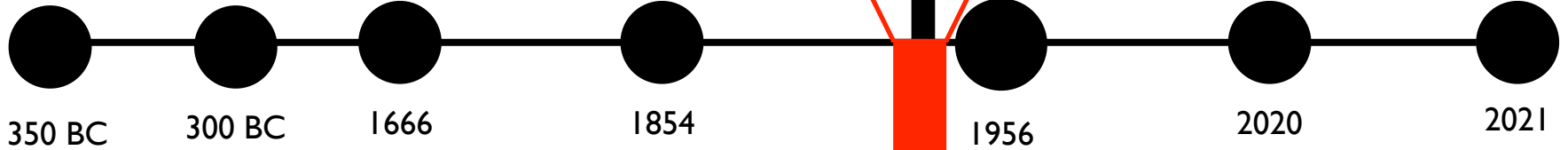
Entscheidungsproblem



“Universal Computational Logic”



Logic Theorist
(birth of modern logicist AI)



350 BC

300 BC

1666

1854

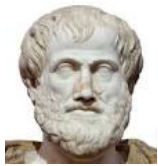
1956

2020

2021



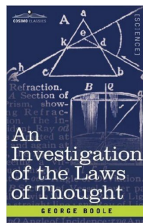
Euclid



Organon



Leibniz



Simon

Intro to (Formal) Logic (& AI)



Frege

T
h
e
S
i
n
g
u
l
a
r
i
t
y
?

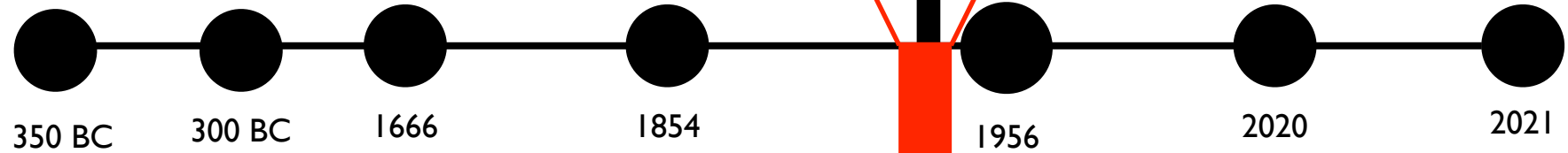
?

Entscheidungsproblem

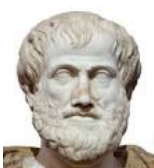
“Universal Computational Logic”



Logic Theorist
(birth of modern logicist AI)



Euclid

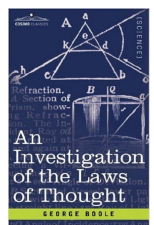


Organon



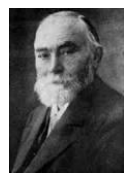
Leibniz

\int



Simon

Intro to (Formal) Logic (& AI)



Frege

Exceeds Leibniz & de-mystifies Euclid: the “compellingness” of these proofs consists in their being, at bottom, formal proofs in first-order logic (FOL).

T
h
e
s
i
n
g
u
l
a
r
i
t
y
?

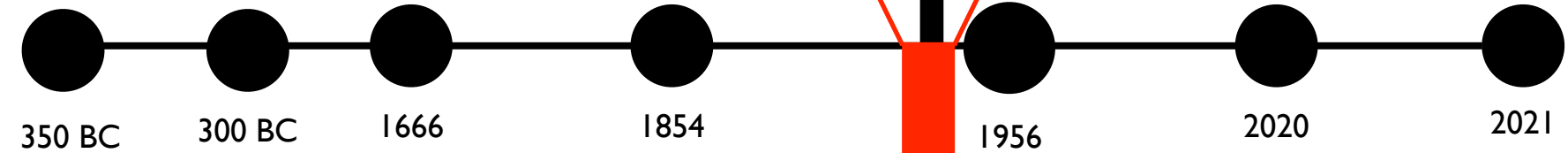
Entscheidungsproblem



“Universal Computational Logic”



Logic Theorist
(birth of modern logicist AI)



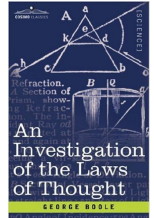
Euclid



Organon



Leibniz



1854



Simon

1956



Frege

Exceeds Leibniz & de-mystifies Euclid: the “compellingness” of these proofs consists in their being, at bottom, formal proofs in first-order logic (FOL).



Church

Intro to (Formal) Logic (& AI)

T
h
e
s
i
n
g
u
l
a
r
i
t
y
?

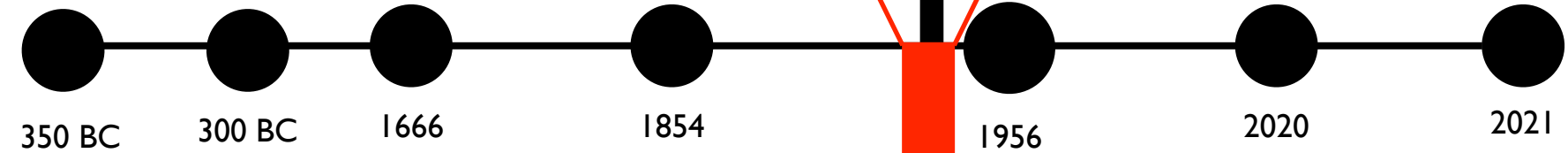
Entscheidungsproblem



“Universal Computational Logic”



Logic Theorist
(birth of modern logicist AI)



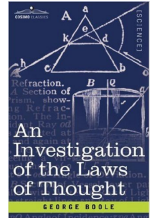
Euclid



Organon



Leibniz



1854



Simon

1956



Frege

Exceeds Leibniz & de-mystifies Euclid: the “compellingness” of these proofs consists in their being, at bottom, formal proofs in first-order logic (FOL).



Church



Turing

2020

2021

Intro to (Formal) Logic (& AI)

T
h
e
S
i
n
g
u
l
a
r
i
t
y
?

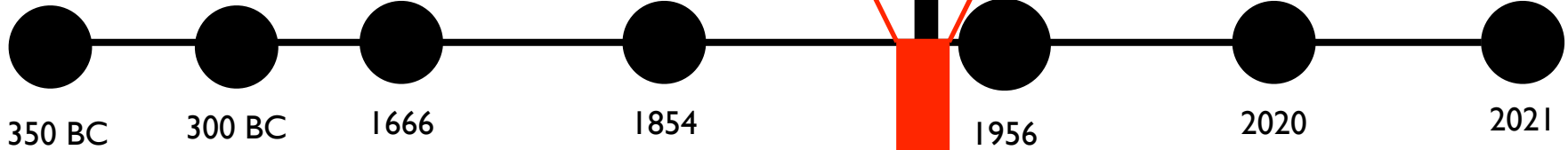
Entscheidungsproblem



“Universal Computational Logic”



Logic Theorist
(birth of modern logicist AI)



350 BC

Euclid



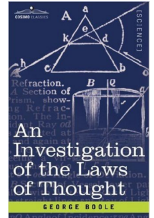
300 BC

Organon



1666

Leibniz



1854

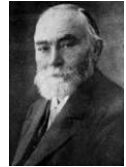


1956

Simon

Intro to (Formal) Logic (& AI)

Exceeds Leibniz & de-mystifies Euclid: the “compellingness” of these proofs consists in their being, at bottom, formal proofs in first-order logic (FOL).



Frege



Church



Turing



Post

T
h
e
s
i
n
g
u
l
a
r
i
t
y
?

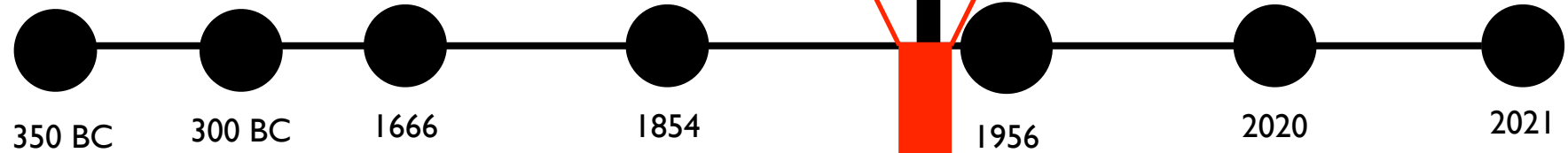
?

Entscheidungsproblem

“Universal Computational Logic”



Logic Theorist
(birth of modern logicist AI)



Euclid

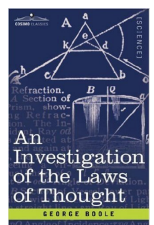


Organon



Leibniz

\int



1854



Simon

1956



Frege

Exceeds Leibniz & de-mystifies Euclid: the “compellingness” of these proofs consists in their being, at bottom, formal proofs in first-order logic (FOL).



Church



Turing



Post

Intro to (Formal) Logic (& AI)

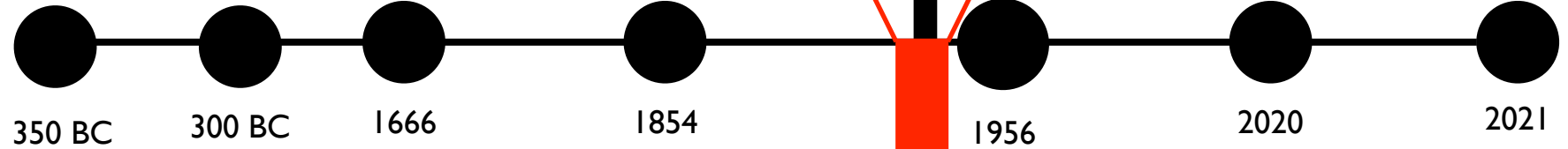
Here’s what a computer is, and given that, sorry, the *Entscheidungsproblem* can’t be solved by such a machine!

T
h
e
s
i
n
g
u
l
a
r
i
t
y
?

New for Today:
 Functional = Church;
 Procedural = Turing.
 Where is logic-based/logicist
 computation/programming?

Entscheidungsproblem

“Universal
 Computational Logic”



Euclid

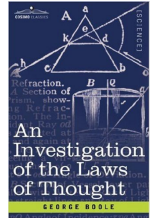


Organon



Leibniz

\int



1854



Simon

Logic Theorist
 (birth of modern logicist AI)

1956

Intro to (Formal) Logic (& AI)



Frege

Exceeds Leibniz & de-mystifies
 Euclid: the “compellingness” of
 these proofs consists in their
 being, at bottom, formal proofs in
 first-order logic (FOL).



Church



Turing



Post

Here’s what a computer is, and
 given that, sorry, the
Entscheidungsproblem can’t be
 solved by such a machine!

T
h
e
s
i
n
g
u
l
a
r
i
t
y
?

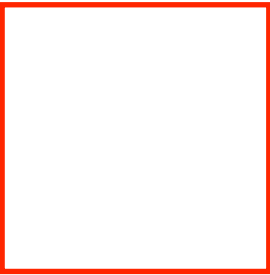
New for Today:
 Functional = Church;
 Procedural = Turing.
 Where is logic-based/logicist
 computation/programming?

“Universal
 Computational Logic”

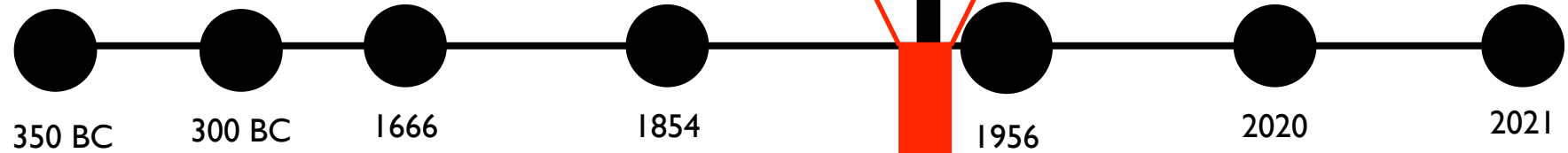


Entscheidungsproblem

?



Logic Theorist
 (birth of modern logicist AI)



Euclid

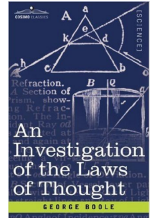


Organon



Leibniz

\int



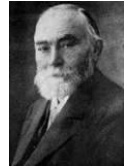
1854



Simon

1956

Intro to (Formal) Logic (& AI)



Frege

Exceeds Leibniz & de-mystifies
 Euclid: the “compellingness” of
 these proofs consists in their
 being, at bottom, formal proofs in
 first-order logic (FOL).



Church



Turing



Post

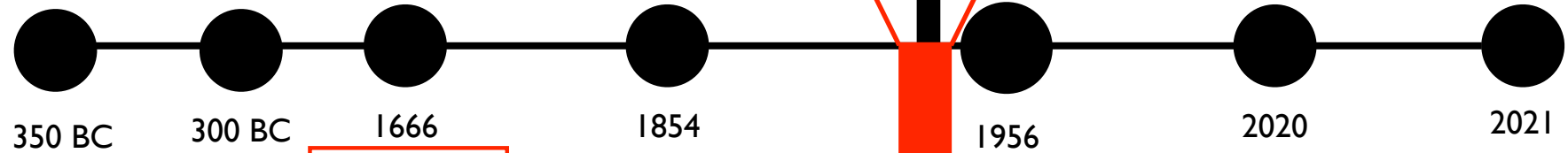
Here’s what a computer is, and
 given that, sorry, the
 Entscheidungsproblem can’t be
 solved by such a machine!

T
h
e
S
i
n
g
u
l
a
r
i
t
y
?

New for Today:
 Functional = Church;
 Procedural = Turing.
 Where is logic-based/logicist
 computation/programming?

Entscheidungsproblem

“Universal
 Computational Logic”



Euclid

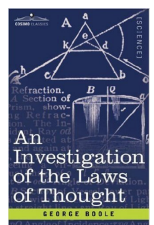


Organon



Leibniz

\int



1854



Simon

Logic Theorist
 (birth of modern logicist AI)

1956

Intro to (Formal) Logic (& AI)



Frege

Exceeds Leibniz & de-mystifies Euclid: the “compellingness” of these proofs consists in their being, at bottom, formal proofs in first-order logic (FOL).



Church



Turing



Post

Here’s what a computer is, and given that, sorry, the Entscheidungsproblem can’t be solved by such a machine!

T h e s i n g u l a r i t y ?

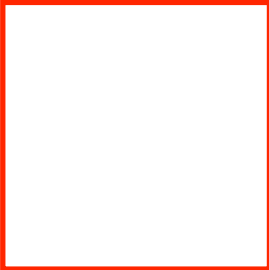
New for Today:
 Functional = Church;
 Procedural = Turing.
 Where is logic-based/logicist
 computation/programming?

“Universal
 Computational Logic”

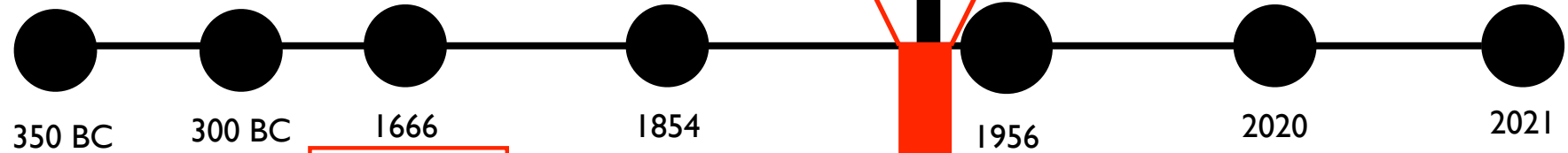


Entscheidungsproblem

?



T
h
e
S
i
n
g
u
l
a
r
i
t
y
?



Euclid

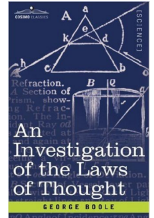


Organon



Leibniz

\int



1854



Simon

1956

Logic Theorist
 (birth of modern logicist AI)

2020

2021

Intro to (Formal) Logic (& AI)



Frege

Exceeds Leibniz & de-mystifies
 Euclid: the “compellingness” of
 these proofs consists in their
 being, at bottom, formal proofs in
 first-order logic (FOL).



Church



Turing



Post

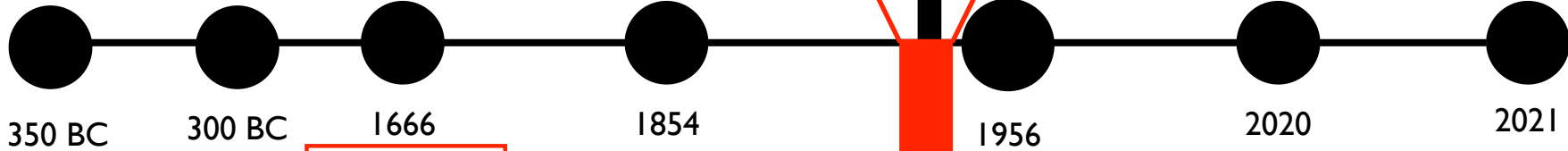
Here’s what a computer is, and
 given that, sorry, the
Entscheidungsproblem can’t be
 solved by such a machine!

New for Today:
Functional = Church;
Procedural = Turing.
Where is logic-based/logicist
computation/programming?

Entscheidungsproblem



“Universal
Computational Logic”



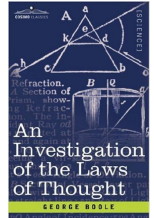
Euclid



Organon



Leibniz



Simon

Intro to (Formal) Logic (& AI)



Frege

Exceeds Leibniz & de-mystifies Euclid: the “compellingness” of these proofs consists in their being, at bottom, formal proofs in first-order logic (FOL).



Church



Turing



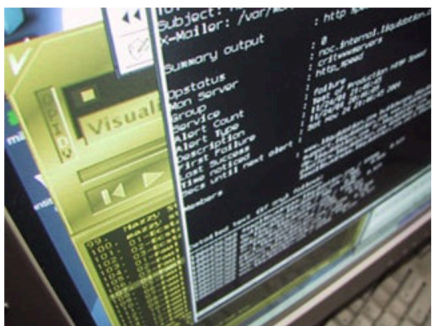
Post

Here’s what a computer is, and given that, sorry, the Entscheidungsproblem can’t be solved by such a machine!

T
h
e
s
i
n
g
u
l
a
r
i
t
y
?

Programming Languages

- COURSE HOME** <
- SYLLABUS
- CALENDAR
- ASSIGNMENTS
- EXAMS
- TOOLS
- DOWNLOAD COURSE MATERIALS



Programming computer screen. (Photo courtesy of [openphoto.net](#).)

Instructor(s)
Prof. Michael Ernst

MIT Course Number
6.821

As Taught In
Fall 2002

Level
Graduate

[CITE THIS COURSE](#)

Course Features

> [Assignments: programming \(no examples\)](#) > [Exams \(no solutions\)](#)

Course Description

6.821 teaches the principles of functional, imperative, and logic programming languages. Topics covered include: meta-circular interpreters, semantics (operational and denotational), type systems (polymorphism, inference, and abstract types), object oriented programming, modules, and multiprocessing. The course involves substantial programming assignments and problem sets as well as a significant amount of reading. The course uses the Scheme+ programming language for all of its assignments.

SYLLABUS

CALENDAR

ASSIGNMENTS

EXAMS

TOOLS

DOWNLOAD COURSE
MATERIALS



MIT Course Number
6.821

As Taught In
Fall 2002

Level
Graduate

CITE THIS COURSE

Programming computer screen. (Photo courtesy of openphoto.net.)

Course Features

> [Assignments: programming \(no examples\)](#) > [Exams \(no solutions\)](#)

Course Description

6.821 teaches the principles of functional, imperative, and logic programming languages. Topics covered include: meta-circular interpreters, semantics (operational and denotational), type systems (polymorphism, inference, and abstract types), object oriented programming, modules, and multiprocessing. The course involves substantial programming assignments and problem sets as well as a significant amount of reading. The course uses the Scheme+ programming language for all of its assignments.

SYLLABUS

CALENDAR

ASSIGNMENTS

EXAMS

TOOLS

DOWNLOAD COURSE MATERIALS



MIT Course Number
6.821

As Taught In
Fall 2002

Level
Graduate

CITE THIS COURSE

Programming computer screen. (Photo courtesy of openphoto.net.)

Course Features

> [Assignments: programming \(no examples\)](#) > [Exams \(no solutions\)](#)

Course Description

6.821 teaches the principles of functional, imperative, and logic programming languages. Topics covered include: meta-circular interpreters, semantics (operational and denotational), type systems (polymorphism, inference, and abstract types), object oriented programming, modules, and multiprocessing. The course involves substantial programming assignments and problem sets as well as a significant amount of reading. The course uses the Scheme+ programming language for all of its assignments.

Syllabus

Programming Languages CSCI-4430

Meetings: Webex, TF 2:30-4:20pm

Website: <http://www.cs.rpi.edu/~milanova/csci4430>

I. Brief Course Description

This course is a study of important concepts in programming languages. Topics include programming language syntax and semantics, types and parameter passing, and programming paradigms (logic-oriented, functional, von Neumann, object-oriented).

Prerequisite: Introduction to Algorithms (CSCI 2300) and Principles of Software (CSCI 2600)

Mailing list: proglang@cs.lists.rpi.edu. Email goes to Milanova, Kuzmin, and Hulbert. Use this list for administrative questions, including homework extension requests, quiz and exam makeup requests, extra time scheduling, and so on.

II. Learning Outcomes

The goal of this course is to teach students how to analyze programming languages. Students will become more productive programmers, will be able to learn new programming languages with ease, and will be able to choose the most suitable programming language for a given problem.

Concretely, students who successfully complete the course should be able to 1) explain programming language syntax and semantics, 2) implement a front-end for a programming language, 3) explain the concepts of scoping, data abstraction, types, control abstraction, and parameter passing, which are essential building blocks of programming languages, and 4) demonstrate competence across a spectrum of programming language paradigms by writing programs in Prolog, Scheme, and Haskell.

III. Required Textbook

Programming Language Pragmatics, Fourth Edition, by Michael Scott, Morgan Kaufmann, 2015.

IV. Class Work and Policies

Quizzes

There are 9 quizzes that should be completed and submitted individually. We will drop the lowest quiz grade and only 8 will count towards the final grade. Quizzes will be administered on Submitly at the beginning of our regularly scheduled class time. We will be offering alternative times for quizzes and exams. **If you are unable to attend during regularly scheduled class hours, you must request an alternative time. Email course staff at proglang@cs.lists.rpi.edu by September 10 outlining the reasons why you will be attending at an alternative time (e.g., you reside in a different time zone). We will assign an alternative time and you will be taking the quizzes during this time slot on the date of the quiz. Note that once assigned, you cannot change the quiz time slot.**

Syllabus

Programming Language

Meetings: **Webex**, TF 2:30-4:20pm

Website: <http://www.cs.rpi.edu/~milanova/csci4430>

I. Brief Course Description

This course is a study of important concepts in programming languages. Topics include programming (logic-oriented, functional, von Neumann, object-oriented).

Prerequisite: Introduction to Algorithms (CSCI 2300) and Principles of Software (CSCI 2600)

Mailing list: proglang@cs.lists.rpi.edu. Email goes to Milanova, Kuzmin, and Hulbert. Use this list for all requests, extra time scheduling, and so on.

II. Learning Outcomes

The goal of this course is to teach students how to analyze programming languages. Students will become more comfortable with programming languages, and will be able to choose the most suitable programming language for a given problem.

Concretely, students who successfully complete the course should be able to 1) explain programming language concepts, 2) explain the concepts of scoping, data abstraction, types, control abstraction, and parameter passing, and 3) apply these concepts across a spectrum of programming language paradigms by writing programs in Prolog, Scheme, and Haskell.

III. Required Textbook

Programming Language Pragmatics, Fourth Edition, by Michael Scott, Morgan Kaufmann, 2015.

IV. Class Work and Policies

Syllabus

Programming Language

Meetings: **Webex**, TF 2:30-4:20pm

Website: <http://www.cs.rpi.edu/~milanova/csci4430>

I. Brief Course Description

This course is a study of important concepts in programming languages. Topics include programming (logic-oriented, functional, von Neumann, object-oriented).

Prerequisite: Introduction to Algorithms (CSCI 2300) and Principles of Software (CSCI 2600)

Mailing list: proglang@cs.lists.rpi.edu. Email goes to Milanova, Kuzmin, and Hulbert. Use this list for a requests, extra time scheduling, and so on.

II. Learning Outcomes

The goal of this course is to teach students how to analyze programming languages. Students will become more comfortable with programming languages, and will be able to choose the most suitable programming language for a given problem.

Concretely, students who successfully complete the course should be able to 1) explain programming concepts, 2) explain the concepts of scoping, data abstraction, types, control abstraction, and parameter passing, and 3) apply these concepts across a spectrum of programming language paradigms by writing programs in Prolog, Scheme, and LISP.

III. Required Textbook

Programming Language Pragmatics, Fourth Edition, by Michael Scott, Morgan Kaufmann, 2015.

IV. Class Work and Policies

There are *Two* Logicist Branches;
BI:

There are *Two* Logicist Branches; B I:

Frege, 1893:

“Aha! Currying! I recast multiple-arity operations with functions into a unary affair!”

There are *Two* Logicist Branches; BI:

Frege, 1893:

“Aha! Currying! I recast multiple-arity operations with functions into a unary affair!”

Schönfinkel, 1920's:

“Aha! I can do this stuff using combinatory logic!”

There are *Two* Logicist Branches; B I:

Frege, 1893:

“Aha! Currying! I recast multiple-arity operations with functions into a unary affair!”

Schönfinkel, 1920's:

“Aha! I can do this stuff using combinatory logic!”

Church, 1920's & 30's:

“Aha! The lambda calculus!”

There are *Two* Logicist Branches; B I:

Frege, 1893:

“Aha! Currying! I recast multiple-arity operations with functions into a unary affair!”

Schönfinkel, 1920's:

“Aha! I can do this stuff using combinatory logic!”

Church, 1920's & 30's:

“Aha! The lambda calculus!

...

There are *Two* Logicist Branches; B I:

Frege, 1893:

“Aha! Currying! I recast multiple-arity operations with functions into a unary affair!”

Schönfinkel, 1920's:

“Aha! I can do this stuff using combinatory logic!”

Church, 1920's & 30's:

“Aha! The lambda calculus!

...

Haskell

There are *Two* Logicist Branches; B I:

Frege, 1893:

“Aha! Currying! I recast multiple-arity operations with functions into a unary affair!”

Schönfinkel, 1920's:

“Aha! I can do this stuff using combinatory logic!”

Church, 1920's & 30's:

“Aha! The lambda calculus!

...

Haskell OCaml, Scheme, ...

There are *Two* Logicist Branches; B I:

Frege, 1893:

“Aha! Currying! I recast multiple-arity operations with functions into a unary affair!”

Schönfinkel, 1920's:

“Aha! I can do this stuff using combinatory logic!”

Church, 1920's & 30's:

“Aha! The lambda calculus!

...

Haskell OCaml, Scheme, ...

Athena

Two Logician Branches; B2:

Two Logician Branches; B2:

The AI Branch: Automated Reasoning

Two Logician Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

Two Logician Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

**Simon & Newell @
Dawn of Modern AI: LT & GPS**

Two Logician Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

**Simon & Newell @
Dawn of Modern AI: LT & GPS**

...

Two Logician Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

**Simon & Newell @
Dawn of Modern AI: LT & GPS**

...

Prolog?

Two Logician Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

**Simon & Newell @
Dawn of Modern AI: LT & GPS**

...

Two Logician Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

**Simon & Newell @
Dawn of Modern AI: LT & GPS**

...

PGLP

Two Logician Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

**Simon & Newell @
Dawn of Modern AI: LT & GPS**

...

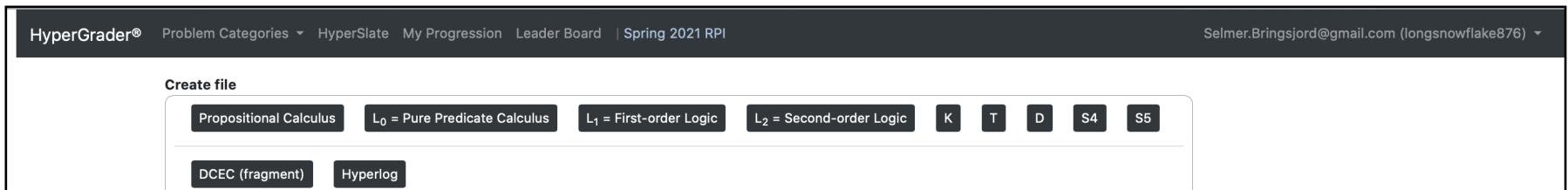
PGLP

Two Logician Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

**Simon & Newell @
Dawn of Modern AI: LT & GPS**



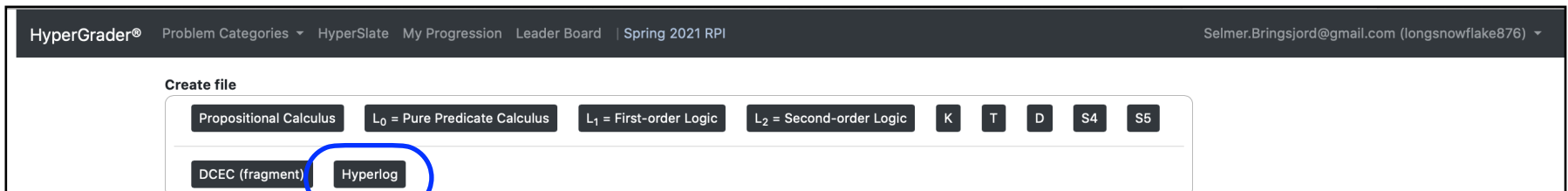
PGLP

Two Logician Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

**Simon & Newell @
Dawn of Modern AI: LT & GPS**



PGLP

Pure General Logic Programming (PGLP)

$$\mathcal{L} := \langle L, \mathcal{I} \rangle$$

$$\frac{\begin{array}{cc} \mathbb{P} & L \\ \mathfrak{q} & L \end{array}}{\mathbb{R} : \langle \mathbb{P}, \mathfrak{q} \rangle \longrightarrow \langle \mathbb{Y} | \mathbb{N} | \mathbb{U}, \delta, \pi_{(s)} | \alpha_{(s)} \rangle}$$

$$\mathbb{C} : \pi_{(s)} | \alpha_{(s)} \longrightarrow \langle \mathbb{Y} | \mathbb{N} | \mathbb{U}, \delta \rangle$$

\mathcal{U}

The Universe of Logics

$DC\mathcal{E}C^*$

$\leq \mathcal{L}_2 =$ second-order logic

$\leq \mathcal{L}_1 =$ first-order logic

$\mathcal{L}_0 =$ zero-order logic

$\mathcal{L}_{pc} \leq \mathcal{L} \leq \mathcal{L}_1 =$ description logics

$\mathcal{L}_{pc} =$ propositional logic

$ID\mathcal{C}E\mathcal{C}$

$\mathcal{L}_{\omega_1\omega}$

Infinitary

Finitary

Intensional

Extensional

Deductive

Inductive

Deductive

Inductive

Symbolic

Diagram

Symbolic

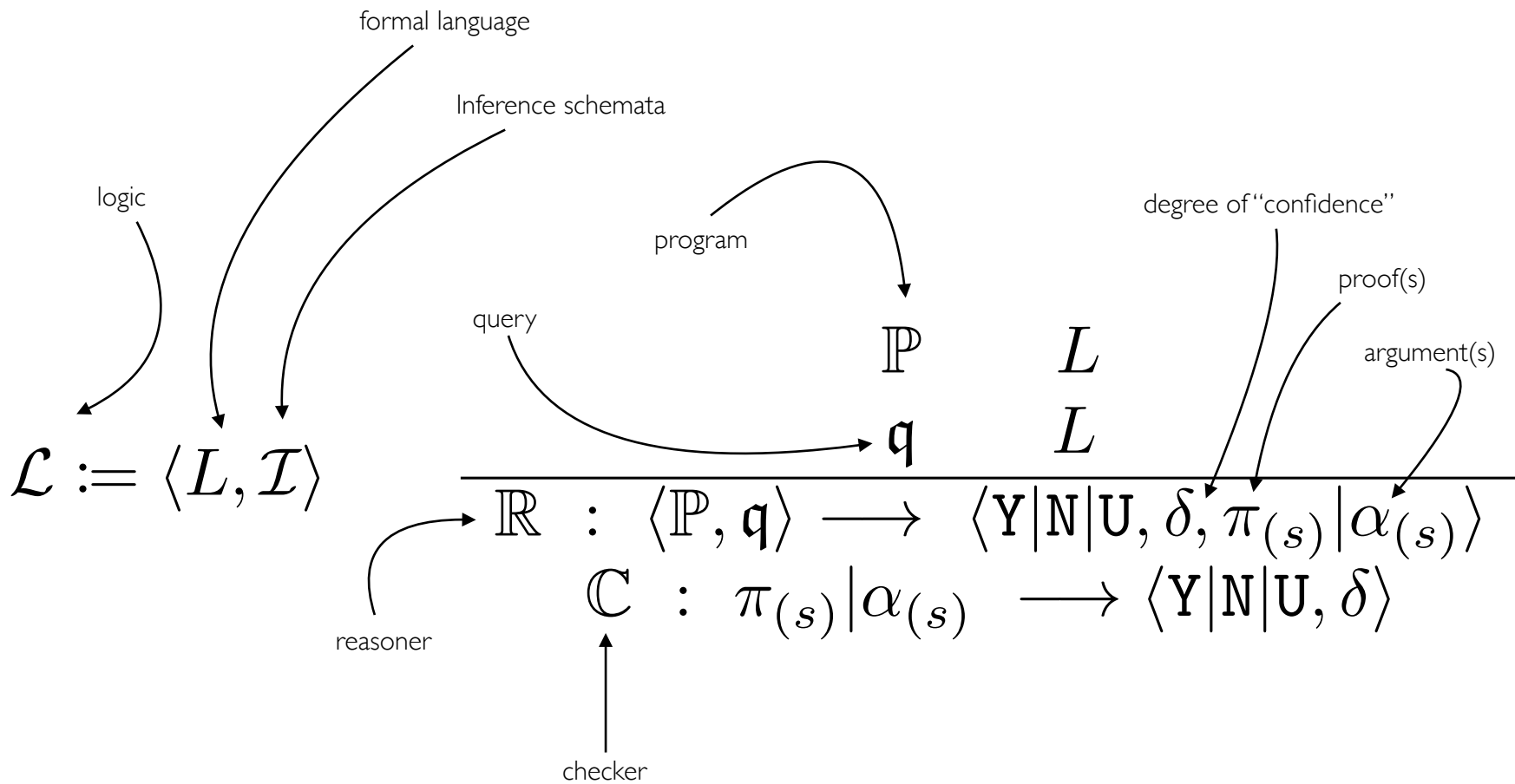
Diagram

Symbolic

Diagram

Symbolic

Diagram



Moronic LLM Content

SB BILLY

Can you create logic programs?



GPT

Yes, I can help you create logic programs. Logic programming involves defining rules and facts to represent knowledge and then using logical inference to derive new information. If you have a specific logic programming language or problem in mind, please let me know, and I'll do my best to assist you.

SB BILLY

Using Aristotle's fragment of first-order logic (from his Organon), create a two-line logic program for answering the query as to whether all the dooopers are feeepy.

Moronic LLM Content

SB BILLY

Can you create

GPT

Yes, I can help
Logic program
and facts to re
using logical in
information. If y
programming l
please let me k
assist you.

SB BILLY

Using Aristotle
(from his Orga
program for an
whether all the

GPT

In Aristotle's fragment of first-order logic, formulation of a two-line logic program can be a bit challenging since his work predates modern logic programming languages. However, I can provide you with a basic logical expression that captures the essence of your query:

1. $\text{dooper}(x) \rightarrow \text{feep}(x)$
2. $\exists x \text{ dooper}(x)$

In this logical program, the first line represents a rule that states if something is a dooper (denoted by $\text{dooper}(x)$), then it is also feep (denoted by $\text{feep}(x)$).

The second line represents the existence of at least one dooper (denoted by $\exists x \text{ dooper}(x)$), which implies that there are entities that satisfy the condition of being a dooper.

Combining these two lines, the logical program implies that all doopers are feep. However, it's important to note that this is a simplified representation based on Aristotle's fragment, and modern logic programming languages may have more explicit syntax and constructs for achieving the same goal.

Moronic LLM Content

GPT

In Aristotle's fragment of first-order logic, formulation of a two-line logic program can be a bit challenging since his work predates modern logic programming languages. However, I can provide you with a basic logical expression that captures the essence of your query:

1. $\text{dooper}(x) \rightarrow \text{feep}(x)$
2. $\exists x \text{ dooper}(x)$

In this logical program, the first line represents a rule that states if something is a dooper (denoted by $\text{dooper}(x)$), then it is feep (denoted by $\text{feep}(x)$).

The second line represents the existence of at least one dooper (denoted by $\exists x \text{ dooper}(x)$), which implies that there are entities that satisfy the condition of being a dooper.

Combining these two lines, the logical program implies that all doopers are feep. However, it's important to note that this is a simplified representation based on Aristotle's fragment, and modern logic programming languages may have more explicit syntax and constructs for achieving the same goal.

BILLY

Using Aristotle's fragment of first-order logic (from his Organon), create a two-line logic program for answering the query as to whether all doopers are feep, such that the answer must be based on reasoning from both lines.

GPT

In Aristotle's fragment of first-order logic, it's not straightforward to create a two-line logic program that provides a conclusive answer based solely on reasoning from those two lines. Aristotle's logic system is limited in expressiveness and lacks some key concepts necessary for such reasoning.

BILLY

Can you create a logic program for answering the query as to whether all the doopers are feep, such that the answer must be based on reasoning from both lines?

GPT

Yes, I can help you create a logic program using Aristotle's fragment of first-order logic and facts to represent the query. The logic program will use logical inference to answer the query based on the information provided. If you have any programming language preferences, please let me know, and I will assist you.

BILLY

Using Aristotle's fragment of first-order logic (from his Organon), create a two-line logic program for answering the query as to whether all the doopers are feep, such that the answer must be based on reasoning from both lines?

Moronic LLM Content

GPT

In Aristotle's fragment of first-order logic, formulation of a two-line logic program can be a bit challenging since his work predates modern logic programming languages. However, I can provide you with a basic logical expression that captures the essence of your query:

1. $\text{dooper}(x) \rightarrow \text{feepey}(x)$
2. $\exists x \text{ dooper}(x)$

In this logical program, the first line represents a rule that states if something is a dooper (denoted by $\text{dooper}(x)$), then it is feepey (denoted by $\text{feepey}(x)$).

The second line represents the assertion that there is at least one dooper (denoted by $\exists x \text{ dooper}(x)$), which implies the existence of entities that satisfy the condition of being a dooper.

Combining these two lines, the logical program implies that all doopers are feepey. However, it's important to note that this is a simplified representation based on Aristotle's fragment, and modern logic programming languages may have more explicit syntax and constructs for achieving the same goal.

BILLY

Using Aristotle's fragment of logic (from his Organon), create a logic program for answering the query: "Whether all doopers are feepey." The answer must be based on the two lines provided.

GPT

In Aristotle's fragment of logic, the first line is not straightforward to create a logic program that provides a conclusive answer based solely on reasoning from the two lines. Aristotle's logic system is not as expressive and lacks the necessary constructs for such reasoning.

However, I can provide a two-line representation that captures the essence of your query while taking into account Aristotle's logic:

1. $\text{dooper}(x) \rightarrow \text{feepey}(x)$
2. $\forall x (\text{dooper}(x) \rightarrow \text{feepey}(x))$

In this logical program, the first line is the same as before, stating that if something is a dooper, then it is feepey.

The second line introduces a universal quantifier ($\forall x$) and strengthens the implication by stating that for all x , if x is a dooper, then it must be feepey. This line represents the assertion that all doopers are feepey.

While these two lines are not sufficient to provide a conclusive answer based solely on reasoning from them, they establish a logical relationship between being a dooper and being feepey according to Aristotle's fragment of first-order logic. Further inference rules and additional premises would be required to draw definitive conclusions.

BILLY

Can you create a logic program for answering the query: "Whether all doopers are feepey." The answer must be based on the two lines provided.

GPT

Yes, I can help you create a logic program for answering the query: "Whether all doopers are feepey." The answer must be based on the two lines provided. I will use logical information and facts to represent the query using logical information. If you have any programming language preferences, please let me know so I can assist you.


BILLY

Using Aristotle's fragment of logic (from his Organon), create a logic program for answering the query: "Whether all the doopers are feepey." The answer must be based on the two lines provided.

$$\mathcal{L}_1 := \langle \mathcal{L}, \mathcal{I} \rangle$$

$$\frac{\begin{array}{l} \forall x(Dx \rightarrow Xx) \\ \mathbb{P} \quad \forall x(Xx \rightarrow Fx) \\ \mathbf{q} \quad \forall x(Dx \rightarrow Fx) \end{array}}{\begin{array}{l} \mathbb{R} : \langle \mathbb{P}, \mathbf{q} \rangle \longrightarrow \langle \mathbf{Y} | \mathbf{N} | \mathbf{U}, \delta, \pi_{(s)} | \alpha_{(s)} \rangle \\ \mathbb{C} : \pi_{(s)} | \alpha_{(s)} \longrightarrow \langle \mathbf{Y} | \mathbf{N} | \mathbf{U}, \delta \rangle \end{array}}$$

A Simple Leibnizian \mathcal{L}_2 Theorem

 Variant1LeibnizsLaw

Here you must prove in second-order logic = \mathcal{L}_2 a consequence of Leibniz's Law (a.k.a. The Identity of Indiscernibles, nicely explained [here](#)), which says that any two things having exactly the same properties are identical, and *vice versa*. Here's the consequence in meta-logic (informally put, that if two things a and b differ with respect to a property, they can't be identical):

$$\exists X(Xa \wedge \neg Xb) \vdash_2 a \neq b$$

We use a subscript of '2' on the provability relation to indicate that the allowable inference schemata in your HyperSlate® are those from *second*-order logic.

You're permitted to leave remaining use of the PC provability oracle in your completed proof. (No deadline for now.)

Moronic LLM Content

SB BILLY

There's a property Arush has but Beegerish lacks, and the latter object is happy.

CHATGPT

It seems like you're describing a situation with two objects or entities, Arush and Beegerish, and attributing properties to them. However, without additional context, it's difficult to provide a meaningful response. Could you provide more details? Are these concepts, people, or perhaps objects in a specific game or story? Please also clarify what you mean by "Beegerish lacks" and "the latter object is happy".

SB BILLY

Do the names 'Arush' and 'Beegerish' denote more than one object?

CHATGPT

Based on the information provided, it's hard to determine if 'Arush' and 'Beegerish' are referring to single or multiple objects. These names could refer to people, concepts, or any number of things depending on context. Could you please provide more information or clarify the context in which these names are being used?

Moronic LLM Content

SB BILLY

There's a property Arush has but Beegerish lacks, and the latter object is happy.

CHATGPT

It seems like you're describing a situation with two objects or entities, Arush and Beegerish, and attributing properties to them. However, without additional context, it's difficult to provide a meaningful response. Could you provide more details? Are these concepts, people, or perhaps objects in a specific game or story? Please also clarify what you mean by "Beegerish lacks" and "latter object is happy".

SB BILLY

Do the names 'Arush' and 'Beegerish' denote more than one object?

CHATGPT

Based on the information provided, it's hard to determine if 'Arush' and 'Beegerish' are referring to single or multiple objects. These names could refer to people, concepts, or any number of things depending on context. Could you please provide more information or clarify the context in which these names are being used?

SB BILLY

They are both as I said objects. You have all the information you need to not only answer the query, but to prove that your answer is correct — assuming sufficient intelligence on your part.

CHATGPT

From the information given, we can infer the following:

Moronic LLM Content

SB BILLY

There's a property Arush has but Beegerish lacks, and the latter object is happy.

CHATGPT

It seems like you're describing a situation with two objects or entities, Arush and Beegerish, and attributing properties to them. However, without additional context, it's difficult to provide a meaningful response.

Could you provide more details? Are these concepts, people, or perhaps objects in a specific game or story? Please also clarify what you mean by "Beegerish lacks" and "latter object is happy".

SB BILLY

Do the names 'Arush' and 'Beegerish' denote more than one object?

CHATGPT

Based on the information provided, it's hard to determine if 'Arush' and 'Beegerish' are referring to single or multiple objects. These names could refer to people, concepts, or any number of things depending on context. Could you please provide more information or clarify the context in which these names are being used?

SB BILLY

They are both as I said objects. You have the information you need to not only answer the query, but to prove that your answer is correct — assuming sufficient intelligence on your part.

CHATGPT

From the information given, we can infer the following:

1. "Arush" is an object with a certain property.
2. "Beegerish" is another object that lacks this property.
3. "Beegerish", the latter object, is happy.

However, we don't have specific details about the properties of these objects. But, given that Beegerish is described as happy, we can infer that the property Arush has and Beegerish lacks could be 'unhappiness' or not being happy. So to answer your question, "Arush" and "Beegerish" do not denote more than one object. Each name refers to one object. The proof lies in the descriptors used: the use of the words "has" and "lacks" suggest singularity, as does the phrase "the latter object," which refers to Beegerish.

$\langle \mathbb{P} \rangle :=$ There's a property Arush has but
Beegerish lacks, and the latter object
is happy.

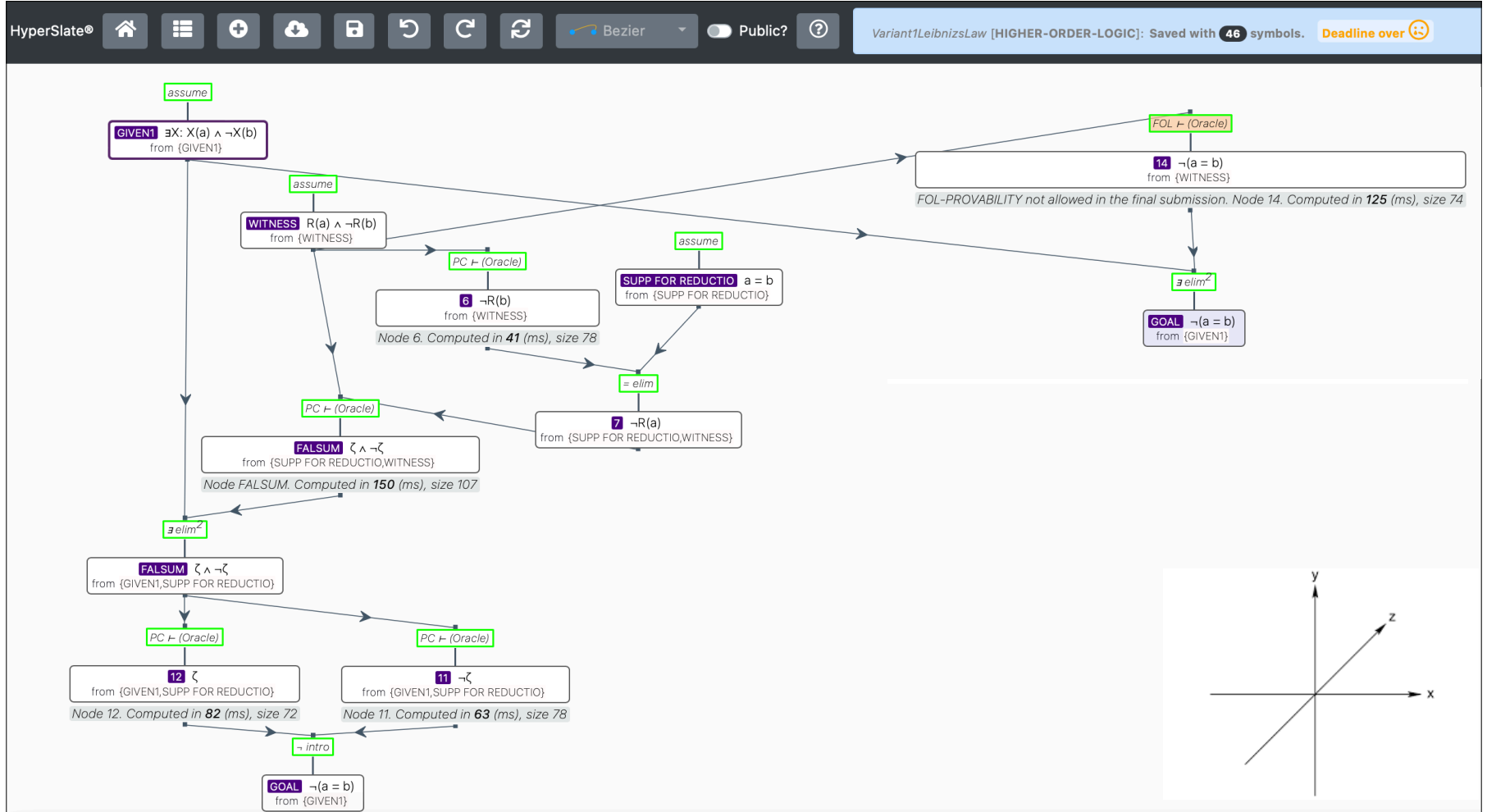
$\langle \mathbf{q} \rangle :=$ Do the names 'Arush' and 'Beegerish'
denote more than one object?

$\mathcal{L} := \langle L, \mathcal{I} \rangle$

$\mathbb{R} : \langle \mathbb{P}, \mathbf{q} \rangle \longrightarrow \langle \mathbf{Y} | \mathbf{N} | \mathbf{U}, \delta, \pi_{(s)} | \alpha_{(s)} \rangle$

$\mathbb{C} : \pi_{(s)} | \alpha_{(s)} \longrightarrow \langle \mathbf{Y} | \mathbf{N} | \mathbf{U}, \delta \rangle$

Execution Produces Answer to Simple Leibnizian Query



Ingredients for Making a PGLP Program ...

On the Anatomy of a PGLP Program

On the Anatomy of a PGLP Program

Linguistics

\vdots	\vdots	\vdots
L_2^μ	meta-level ₂ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level ₁ language	$\exists x \text{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{A} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

On the Anatomy of a PGLP Program

Linguistics

\vdots	\vdots	\vdots
L_2^μ	meta-level ₂ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level ₁ language	$\exists x \text{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{A} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of inference schemata. (For economy, see coming Example 1.)

On the Anatomy of a PGLP Program

Linguistics

\vdots	\vdots	\vdots
L_2^μ	meta-level ₂ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level ₁ language	$\exists x \text{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{A} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of inference schemata. (For economy, see coming Example 1.)

Semantics

Reasoning-semantic; wholly inferentialist (after all, what's the semantics of deduction over meta-level₁ formulae??).

On the Anatomy of a PGLP Program

Linguistics

\vdots	\vdots	\vdots
L_2^μ	meta-level ₂ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level ₁ language	$\exists x \text{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{A} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of inference schemata. (For economy, see coming Example 1.)

Semantics

Reasoning-semantic; wholly inferentialist (after all, what's the semantics of deduction over meta-level₁ formulae??).

On the Anatomy of a PGLP Program

Linguistics

\vdots	\vdots	\vdots
L_2^μ	meta-level ₂ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level ₁ language	$\exists x \text{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{A} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of inference schemata. (For economy, see coming Example 1.)

Semantics

Reasoning-semantic; wholly inferentialist (after all, what's the semantics of deduction over meta-level₁ formulae??).

\mathcal{L}

On the Anatomy of a PGLP Program

Linguistics

\vdots	\vdots	\vdots
L_2^μ	meta-level ₂ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level ₁ language	$\exists x \text{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{A} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of inference schemata. (For economy, see coming Example 1.)

Semantics

Reasoning-semantic; wholly inferentialist (after all, what's the semantics of deduction over meta-level₁ formulae??).

\mathcal{L}

Selection of language, inference schemata, plus formulae/meta-formulae = $\mathbb{P}_{\mathcal{L}}$

On the Anatomy of a PGLP Program

Linguistics

\vdots	\vdots	\vdots
L_2^μ	meta-level ₂ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level ₁ language	$\exists x \text{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{A} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of inference schemata. (For economy, see coming Example 1.)

Semantics

Reasoning-semantic; wholly inferentialist (after all, what's the semantics of deduction over meta-level₁ formulae??).

\mathcal{L}

On the Anatomy of a PGLP Program

Linguistics

\vdots	\vdots	\vdots
L_2^μ	meta-level ₂ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level ₁ language	$\exists x \text{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{A} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of inference schemata. (For economy, see coming Example 1.)

Semantics

Reasoning-semantic; wholly inferentialist (after all, what's the semantics of deduction over meta-level₁ formulae??).

\mathcal{L}

On the Anatomy of a PGLP Program

Linguistics

\vdots	\vdots	\vdots
L_2^μ	meta-level ₂ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level ₁ language	$\exists x \text{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{A} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of inference schemata. (For economy, see coming Example 1.)

Semantics

Reasoning-semantic; wholly inferentialist (after all, what's the semantics of deduction over meta-level₁ formulae??).

\mathcal{L}

Slutten

Slutten

Part II:

**Review of All Inference Rules/
Schemata in PropCalc = \mathcal{L}_{PC}**