Tutorial, Mechanics; Historical & Scientific Context re Formal Logic, AI, and Logic Machines

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

Intermediate Formal Logic & AI (IFLAI2)
9/3/2020

Mechanics …

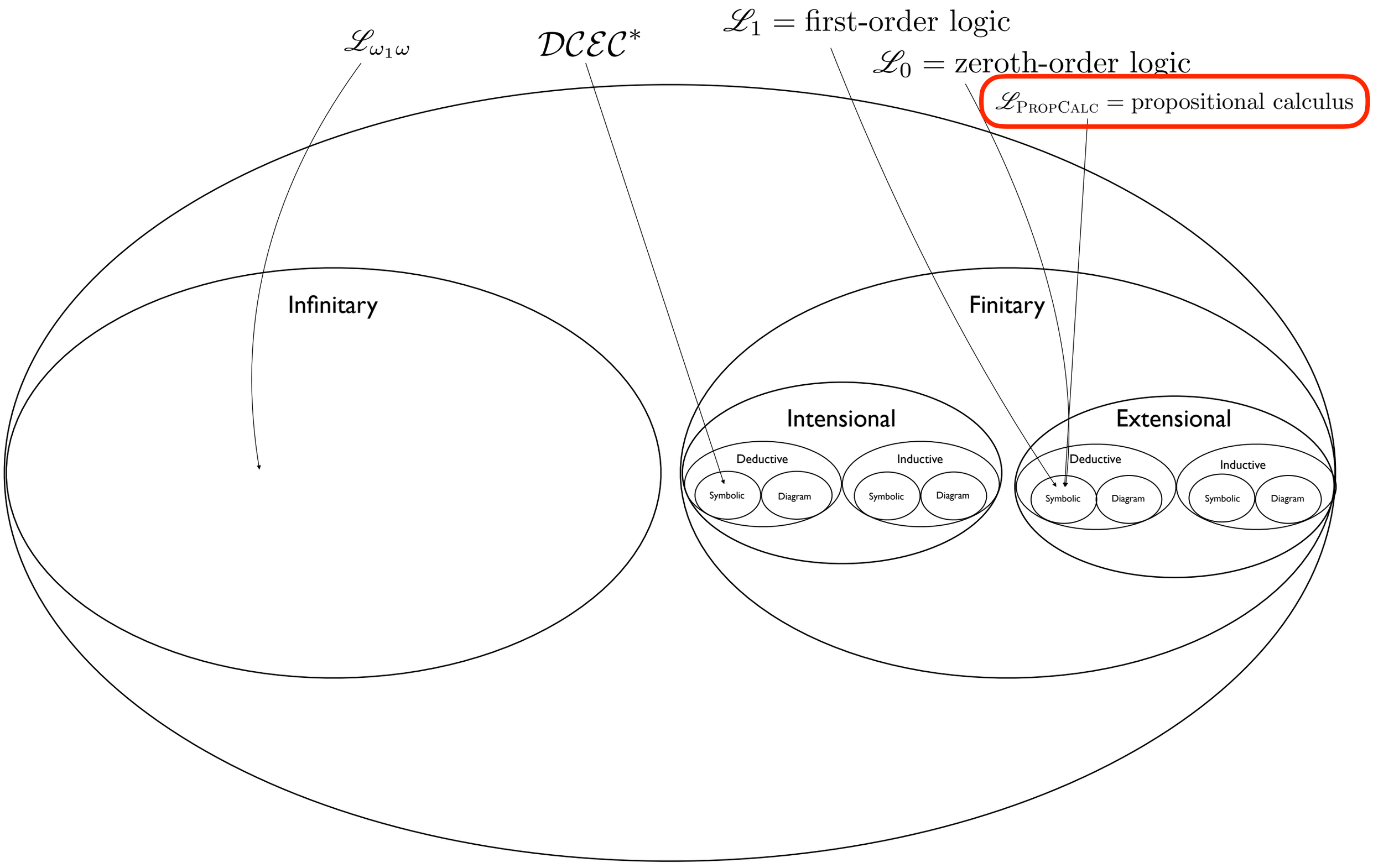
Mechanics …

Mechanics …

IFLAI2 web page now page up.

Some Review/ Diagnostics …

The Universe of Logics



A criminal genius nearly a match for Sherlock Holmes (Do you recognize the Dr?) has built a massive hydrogen bomb, and life on Earth is hanging in the balance, hinging on whether you make the logical prediction. Dr M gives you a sporting chance to: make the right prediction, snip or not snip accordingly, and prove that you're right …

A criminal genius nearly a match for Sherlock Holmes (Do you recognize the Dr?)

A criminal genius nearly a match for Sherlock Holmes (Do you recognize the Dr?) has built a massive hydrogen bomb, and life on Earth is hanging in the balance, hinging on whether you make the logical prediction. Dr M gives you a sporting chance to: make the right prediction, snip or not snip accordingly, and prove that you're right …

If one of the following assertions is true then so is the other:

(1) If the red wire runs to the bomb, then the blue wire runs to the bomb; and, if the blue wire runs to the bomb, then the red wire runs to the bomb.

(2) The red wire runs to the bomb.

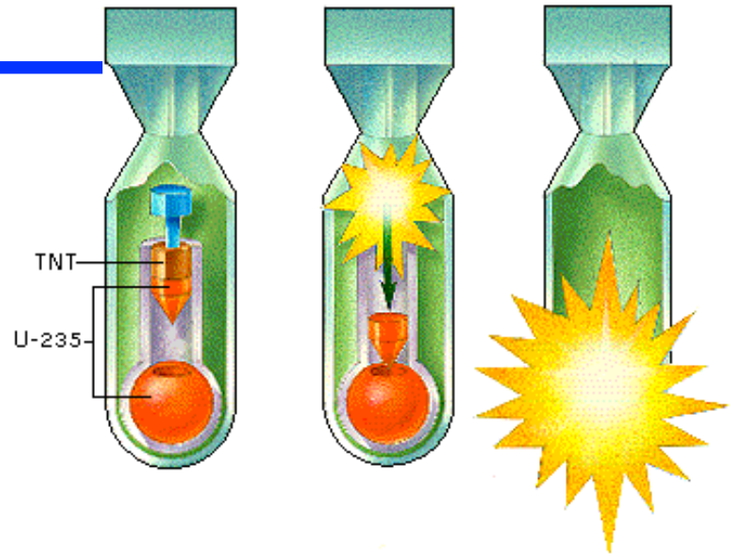
Given this perfectly reliable clue from Dr Moriarty, if either wire is more likely to run to the bomb, that wire *do*es run to the bomb, and the bomb is ticking, with only a minute left! If both are equiprobable, neither runs to the bomb, and you are powerless. Make your prediction as to what will happen when a wire is snipped, and then make your selected snip by clicking on the wire you want to snip! Or leave well enough alone!

Red more likely.

Blue more likely.

Equiprobable.

Snip



Life on Earth has ended.

advance one more slide to see a proof that you indeed made an irrational decision…

Proposition: The blue wire is more likely!

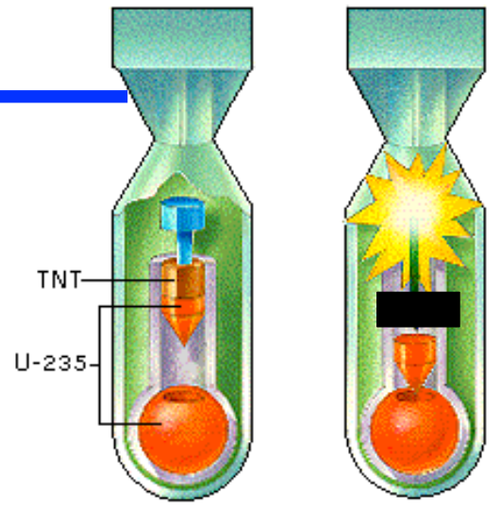
Proof: (1) can be treated as a biconditional, obviously (R <=> B).

There are two top-level cases to consider: (1) and (2) are both true; or both are false. In the case where they are both true, it's trivial to deduce both R and B. So far, then, R and B are equiprobable. What happens in the case where (1) and (2) are both false? We immediately have ~R from the denial of (2). But a biconditional is true just in case both sides are true, or both sides are false; so we have two sub-cases to consider.

Consider first the case where R is true and B is false. We have an immediate contradiction in this sub-case, so both R and B can both be deduced here, and we have not yet departed from equiprobable. So what about the case where R is false and B is true? The falsity of R is not new information (we already have that from the denial of (2)), but we can still derive B. Hence the blue wire is more likely. **QED**

STOP

Snip



Life on Earth is saved!

if you can now hand Dr M a proof that your decision was the rational one!

Advance one more slide to see a proof from Bringsjord that yours had better match up to …

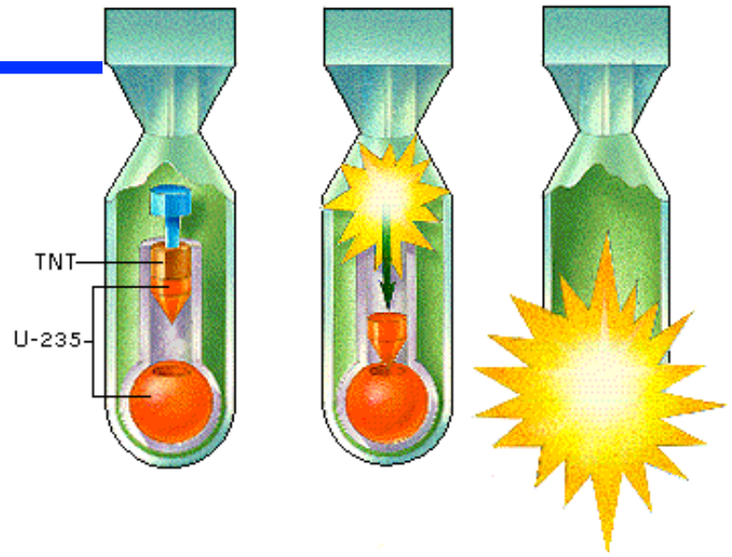
Proposition: The blue wire is more likely!

Proof: (1) can be treated as a biconditional, obviously (R <=> B).

There are two top-level cases to consider: (1) and (2) are both true; or both are false. In the case where they are both true, it's trivial to deduce both R and B. So far, then, R and B are equiprobable. What happens in the case where (1) and (2) are both false? We immediately have ~R from the denial of (2). But a biconditional is true just in case both sides are true, or both sides are false; so we have two sub-cases to consider.

Consider first the case where R is true and B is false. We have an immediate contradiction in this sub-case, so both R and B can both be deduced here, and we have not yet departed from equiprobable. So what about the case where R is false and B is true? The falsity of R is not new information (we already have that from the denial of (2)), but we can still derive B. Hence the blue wire is more likely. **QED**

STOP



Life on Earth has ended.

advance one more slide to see a proof that you indeed made an irrational decision…

Proposition: The blue wire is more likely!

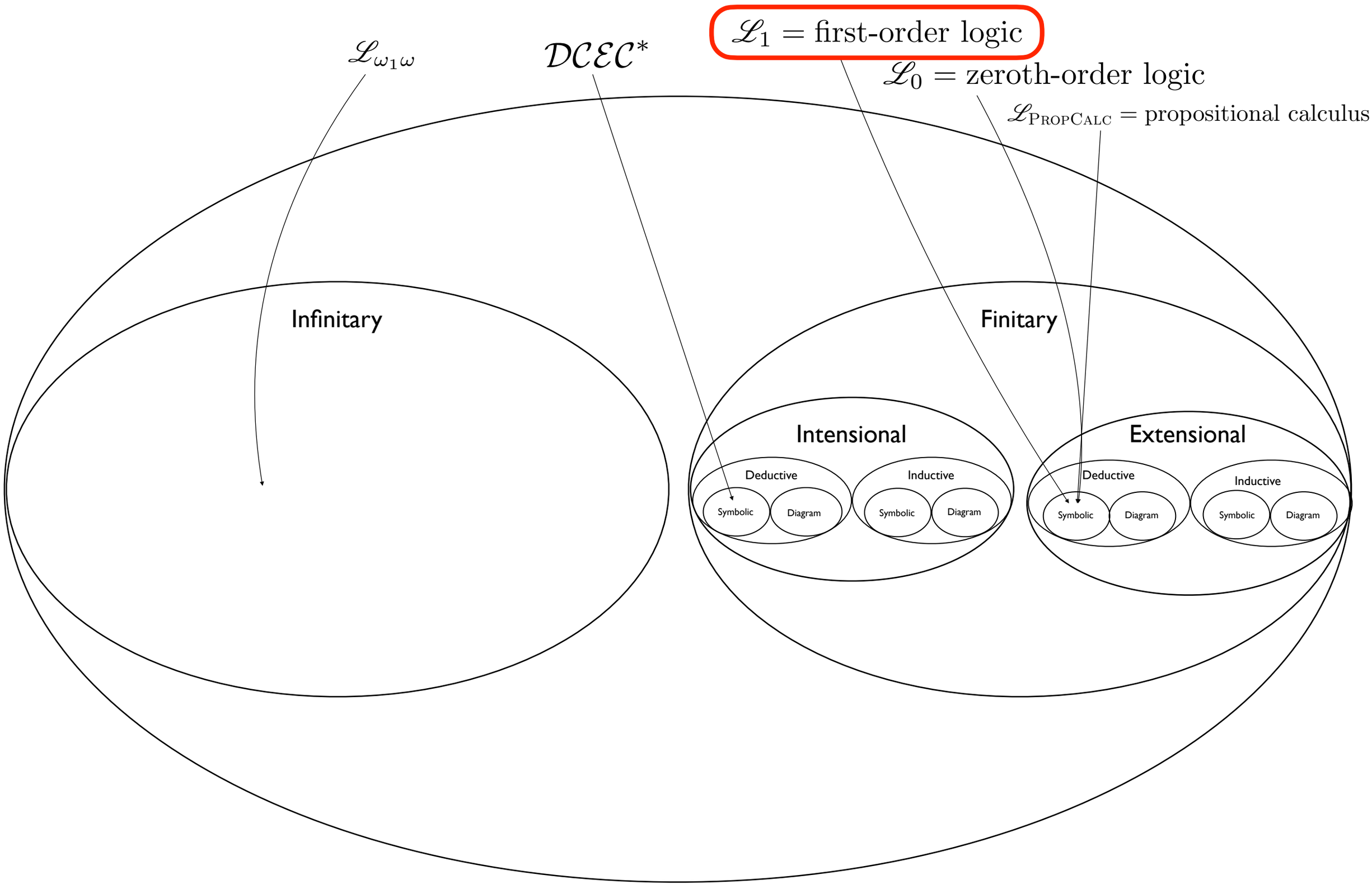
Proof: (1) can be treated as a biconditional, obviously (R <=> B).

There are two top-level cases to consider: (1) and (2) are both true; or both are false. In the case where they are both true, it's trivial to deduce both R and B. So far, then, R and B are equiprobable. What happens in the case where (1) and (2) are both false? We immediately have ~R from the denial of (2). But a biconditional is true just in case both sides are true, or both sides are false; so we have two sub-cases to consider.

Consider first the case where R is true and B is false. We have an immediate contradiction in this sub-case, so both R and B can both be deduced here, and we have not yet departed from equiprobable. So what about the case where R is false and B is true? The falsity of R is not new information (we already have that from the denial of (2)), but we can still derive B. Hence the blue wire is more likely. **QED**

STOP

The Universe of Logics



Special Llamas Disjunction

There's a thing such that it's both a llama and a non-llama;
or
there's a thing such that if it's a llama, everything is a llama;
or
there's a thing such that every llama is a non-llama.

Special Llamas Disjunction

There's a thing such that it's both a llama and a non-llama;
or
there's a thing such that if it's a llama, everything is a llama;
or
there's a thing such that every llama is a non-llama.

Is this disjunction TRUE, FALSE, or UNKNOWN?

Special Llamas Disjunction

There's a thing such that it's both a llama and a non-llama;
or
there's a thing such that if it's a llama, everything is a llama;
or
there's a thing such that every llama is a non-llama.

Is this disjunction TRUE, FALSE, or UNKNOWN?

(Can you (later, if catching up) build a formal, verifying proof in HyperSlate®!)

Special Llamas Disjunction

There's a thing such that it's both a llama and a non-llama;
or
there's a thing such that if it's a llama, everything is a llama;
or
there's a thing such that every llama is a non-llama.

Is this disjunction TRUE, FALSE, or UNKNOWN?

(Can you (later, if catching up) build a formal, verifying proof in HyperSlate®!)

Background Claim

$\mathcal{R}$ Humans, at least neurobiologically normal ones, are fundamentally rational, where rationality is constituted by certain logico-mathematically based reasoning and decision-making in response to real-world stimuli, including stimuli given in the form of focused tests; but mere animals are not fundamentally rational, since, *contra* Darwin, their minds are fundamentally qualitatively inferior to the human mind. As to whether computing machines/robots are fundamentally rational, the answer is "No." For starters, if x can't read, write, and create, x can't be rational; computing machines/robots can neither read nor write nor create; ergo, they aren't fundamentally rational.

Background Claim

$\mathcal{R}$ Humans, at least neurobiologically normal ones, are fundamentally rational, where rationality is constituted by certain logico-mathematically based reasoning and decision-making in response to real-world stimuli, including stimuli given in the form of focused tests; but mere animals are not fundamentally rational, since, *contra* Darwin, their minds are fundamentally qualitatively inferior to the human mind. As to whether computing machines/robots are fundamentally rational, the answer is "No." For starters, if x can't read, write, and create, x can't be rational; computing machines/robots can neither read nor write nor create; ergo, they aren't fundamentally rational.

To infinity and beyond! — routinely

Background Claim

$\mathcal{R}$ Humans, at least neurobiologically normal ones, are fundamentally rational, where rationality is constituted by certain logico-mathematically based reasoning and decision-making in response to real-world stimuli, including stimuli given in the form of focused tests; but mere animals are not fundamentally rational, since, *contra* Darwin, their minds are fundamentally qualitatively inferior to the human mind. As to whether computing machines/robots are fundamentally rational, the answer is "No." For starters, if x can't read, write, and create, x can't be rational; computing machines/robots can neither read nor write nor create; ergo, they aren't fundamentally rational.

self-reference

To infinity and beyond! — routinely

Background Claim

$\mathcal{R}$ Humans, at least neurobiologically normal ones, are fundamentally rational, where rationality is constituted by certain logico-mathematically based reasoning and decision-making in response to real-world stimuli, including stimuli given in the form of focused tests; but mere animals are not fundamentally rational, since, *contra* Darwin, their minds are fundamentally qualitatively inferior to the human mind. As to whether computing machines/robots are fundamentally rational, the answer is "No." For starters, if x is fundamentally irrational, x can't be rational; computing machines/robots can neither read nor write nor create; ergo, they aren't fundamentally rational.

recursion

self-reference

To infinity and beyond! — routinely

Background Claim

intensional reasoning

recursion

self-reference

To infinity and beyond! — routinely

quantification

intensional reasoning

recursion

self-reference

To infinity and beyond! — routinely

abstract-and-valid inference schemata

quantification

intensional reasoning

recursion

self-reference

To infinity and beyond! — routinely

abstract-and-valid inference schemata

quantification

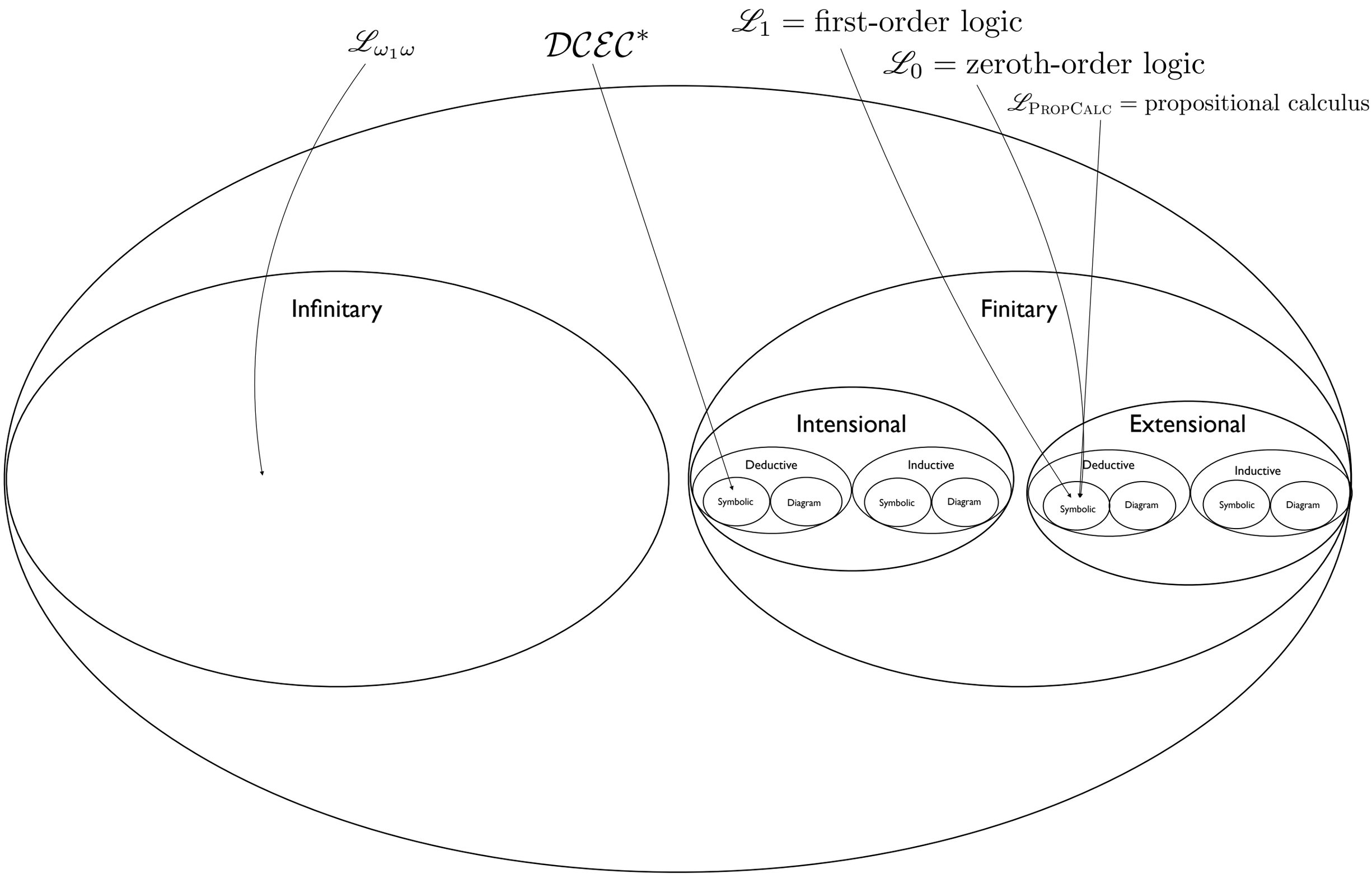
intensional reasoning

recursion

self-reference

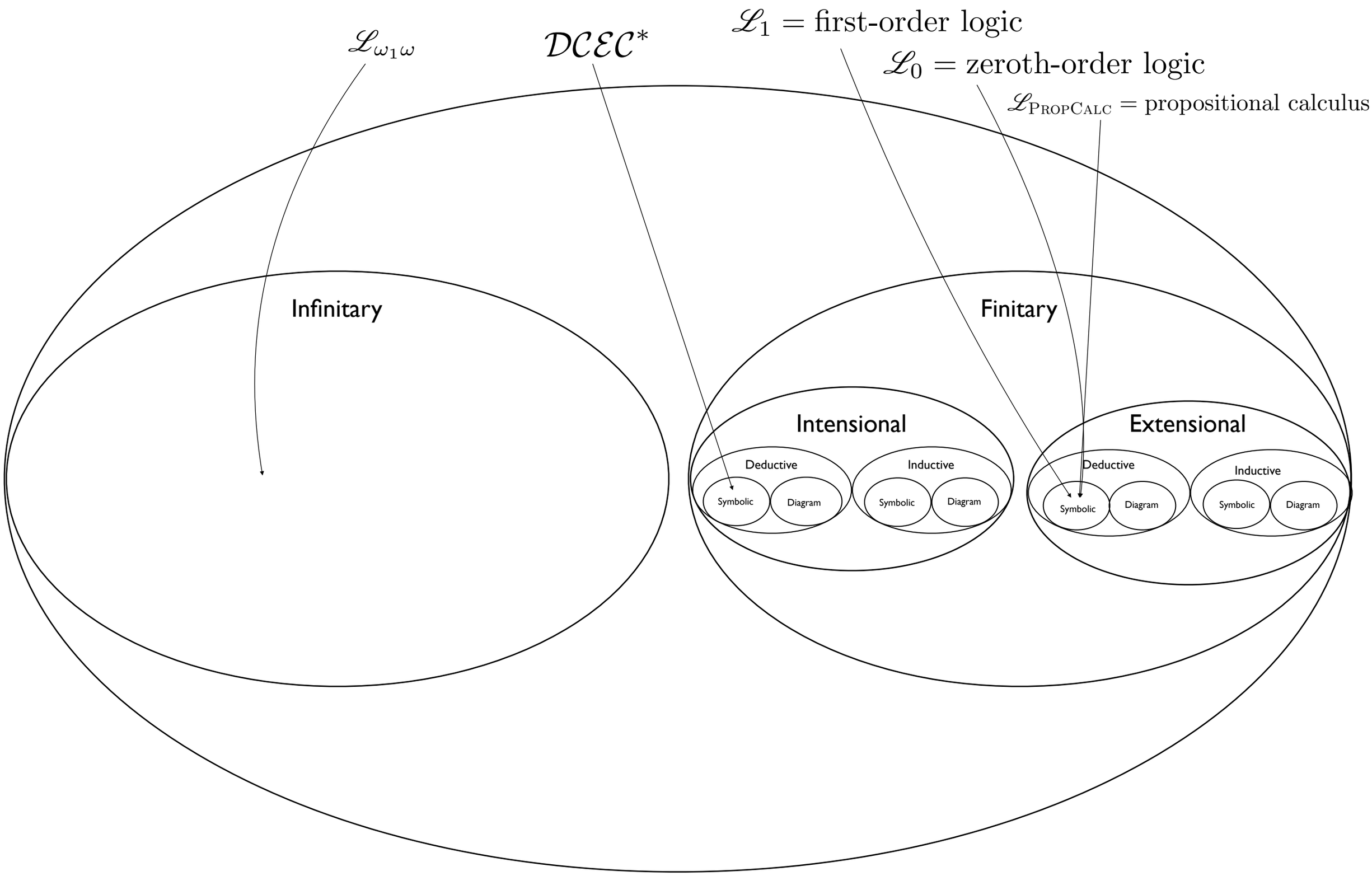
To infinity and beyond! — routinely

The Universe of Logics

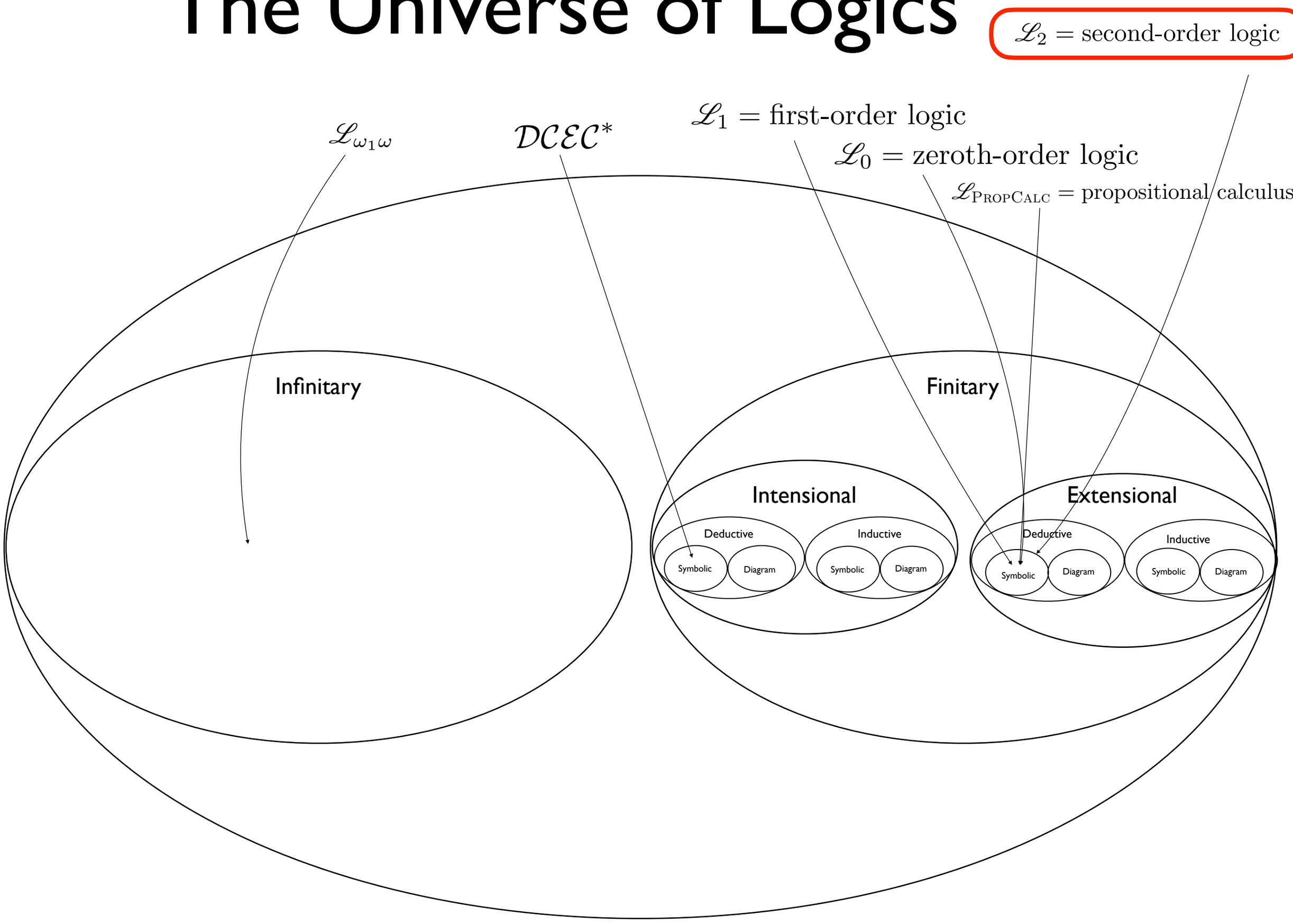


The Universe of Logics

$\mathscr{L}_2$ = second-order logic



The Universe of Logics



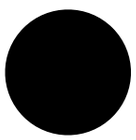
3 HG® Problems

- In addition to switching_conjuncts_fine, used for the signin tutorial, there are now 3 problem due Tues Sep 8:
  - RipsSaysNo2 ($\mathscr{L}_{pc}$)
  - OnlyMediumOrLargeLlamas ($\mathscr{L}_1$)
  - DoubleMindedMan ($\mathscr{L}_2$)

An Anchoring Timeline

• • •

2020



2020

IFLAI2 @ RPI

$\mathcal{DCEC}^*$

Syntax

$$
\begin{aligned}
S ::= {} & \text{Object} \mid \text{Agent} \mid \text{Self} \sqsubset \text{Agent} \mid \text{ActionType} \mid \text{Action} \sqsubseteq \text{Event} \mid \\
& \text{Moment} \mid \text{Boolean} \mid \text{Fluent} \mid \text{Numeric} \\
\\
& \mathit{action} : \text{Agent} \times \text{ActionType} \rightarrow \text{Action} \\
& \mathit{initially} : \text{Fluent} \rightarrow \text{Boolean} \\
& \mathit{holds} : \text{Fluent} \times \text{Moment} \rightarrow \text{Boolean} \\
& \mathit{happens} : \text{Event} \times \text{Moment} \rightarrow \text{Boolean} \\
& \mathit{clipped} : \text{Moment} \times \text{Fluent} \times \text{Moment} \rightarrow \mathit{Boolean} \\
f ::= {} & \mathit{initiates} : \text{Event} \times \text{Fluent} \times \text{Moment} \rightarrow \text{Boolean} \\
& \mathit{terminates} : \text{Event} \times \text{Fluent} \times \text{Moment} \rightarrow \text{Boolean} \\
& \mathit{prior} : \text{Moment} \times \text{Moment} \rightarrow \text{Boolean} \\
& \mathit{interval} : \text{Moment} \times \text{Boolean} \\
& * : \text{Agent} \rightarrow \text{Self} \\
& \mathit{payoff} : \text{Agent} \times \text{ActionType} \times \text{Moment} \rightarrow \text{Numeric} \\
\\
t ::= {} & x : S \mid c : S \mid f(t_1, \ldots, t_n) \\
\\
& t : \text{Boolean} \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \\
& \mathbf{P}(a,t,\phi) \mid \mathbf{K}(a,t,\phi) \mid \mathbf{C}(t,\phi) \mid \mathbf{S}(a,b,t,\phi) \mid \mathbf{S}(a,t,\phi) \\
\phi ::= {} & \mathbf{B}(a,t,\phi) \mid \mathbf{D}(a,t,holds(f,t')) \mid \mathbf{I}(a,t,happens(action(a^*,\alpha),t')) \\
& \mathbf{O}(a,t,\phi,happens(action(a^*,\alpha),t'))
\end{aligned}
$$

Rules of Inference

$$\frac{}{\mathbf{C}(t,\mathbf{P}(a,t,\phi) \rightarrow \mathbf{K}(a,t,\phi))}\ [R_1] \qquad \frac{}{\mathbf{C}(t,\mathbf{K}(a,t,\phi) \rightarrow \mathbf{B}(a,t,\phi))}\ [R_2]$$

$$\frac{\mathbf{C}(t,\phi)\ t \leq t_1 \ldots t \leq t_n}{\mathbf{K}(a_1,t_1,\ldots\mathbf{K}(a_n,t_n,\phi)\ldots)}\ [R_3] \qquad \frac{\mathbf{K}(a,t,\phi)}{\phi}\ [R_4]$$

$$\frac{}{\mathbf{C}(t,\mathbf{K}(a,t_1,\phi_1 \rightarrow \phi_2)) \rightarrow \mathbf{K}(a,t_2,\phi_1) \rightarrow \mathbf{K}(a,t_3,\phi_2)}\ [R_5]$$

$$\frac{}{\mathbf{C}(t,\mathbf{B}(a,t_1,\phi_1 \rightarrow \phi_2)) \rightarrow \mathbf{B}(a,t_2,\phi_1) \rightarrow \mathbf{B}(a,t_3,\phi_2)}\ [R_6]$$

$$\frac{}{\mathbf{C}(t,\mathbf{C}(t_1,\phi_1 \rightarrow \phi_2)) \rightarrow \mathbf{C}(t_2,\phi_1) \rightarrow \mathbf{C}(t_3,\phi_2)}\ [R_7]$$

$$\frac{}{\mathbf{C}(t,\forall x.\ \phi \rightarrow \phi[x \mapsto t])}\ [R_8] \qquad \frac{}{\mathbf{C}(t,\phi_1 \leftrightarrow \phi_2 \rightarrow \neg\phi_2 \rightarrow \neg\phi_1)}\ [R_9]$$

$$\frac{}{\mathbf{C}(t,[\phi_1 \wedge \ldots \wedge \phi_n \rightarrow \phi] \rightarrow [\phi_1 \rightarrow \ldots \rightarrow \phi_n \rightarrow \psi])}\ [R_{10}]$$

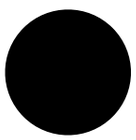
$$\frac{\mathbf{B}(a,t,\phi)\ \ \phi \rightarrow \psi}{\mathbf{B}(a,t,\psi)}\ [R_{11a}] \qquad \frac{\mathbf{B}(a,t,\phi)\ \ \mathbf{B}(a,t,\psi)}{\mathbf{B}(a,t,\psi \wedge \phi)}\ [R_{11b}]$$

$$\frac{\mathbf{S}(s,h,t,\phi)}{\mathbf{B}(h,t,\mathbf{B}(s,t,\phi))}\ [R_{12}]$$

$$\frac{\mathbf{I}(a,t,happens(action(a^*,\alpha),t'))}{\mathbf{P}(a,t,happens(action(a^*,\alpha),t))}\ [R_{13}]$$

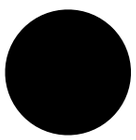
$$\frac{\begin{array}{l}\mathbf{B}(a,t,\phi)\ \ \mathbf{B}(a,t,\mathbf{O}(a^*,t,\phi,happens(action(a^*,\alpha),t'))) \\ \mathbf{O}(a,t,\phi,happens(action(a^*,\alpha),t'))\end{array}}{\mathbf{K}(a,t,\mathbf{I}(a^*,t,happens(action(a^*,\alpha),t')))}\ [R_{14}]$$

$$\frac{\phi \leftrightarrow \psi}{\mathbf{O}(a,t,\phi,\gamma) \leftrightarrow \mathbf{O}(a,t,\psi,\gamma)}\ [R_{15}]$$



2020

IFLAI2 @ RPI



2020

IFLAI2 @ RPI

2020

IFLAI2 @ RPI

2020

IFLAI2 @ RPI

350 BC

2020

IFLAI2 @ RPI

350 BC

Euclid

2020

IFLAI2 @ RPI

Euclidean "Magic"

Theorem: There are infinitely many primes.

Proof: We take an indirect route. Let $\Pi = p_1 = 2, p_2 = 3, p_3 = 5, \ldots, p_k$ be a finite, exhaustive consecutive sequence of prime numbers. Next, let $\mathbf{M}_\Pi$ be $p_1 \times p_2 \times \cdots \times p_k$, and set $\mathbf{M}'_\Pi$ to $\mathbf{M}_\Pi + 1$. Either $\mathbf{M}'_\Pi$ is prime, or not; we thus have two (exhaustive) cases to consider.

C1 Suppose $\mathbf{M}'_\Pi$ is prime. In this case we immediately have a prime number beyond any in Π — contradiction!

C2 Suppose on the other hand that $\mathbf{M}'_\Pi$ is *not* prime. Then some prime p divides $\mathbf{M}'_\Pi$. (Why?) Now, p itself is either in Π, or not; we hence have two sub-cases. Supposing that p is in Π entails that p divides $\mathbf{M}_\Pi$. But we are operating under the supposition that p divides $\mathbf{M}'_\Pi$ as well. This implies that p divides 1, which is absurd (a contradiction). Hence the prime p is outside Π.

Hence for *any* such list Π, there is a prime outside the list. That is, there are infinitely many primes. **QED**

Euclidean "Magic"

Theorem: There are infinitely many primes.

Proof: We take an indirect route. Let $\Pi = p_1 = 2, p_2 = 3, p_3 = 5, \ldots, p_k$ be a finite, exhaustive consecutive sequence of prime numbers. Next, let $\mathbf{M}_\Pi$ be $p_1 \times p_2 \times \cdots \times p_k$, and set $\mathbf{M}'_\Pi$ to $\mathbf{M}_\Pi + 1$. Either $\mathbf{M}'_\Pi$ is prime, or not; we thus have two (exhaustive) cases to consider.

C1 Suppose $\mathbf{M}'_\Pi$ is prime. In this case we immediately have a prime number beyond any in Π — contradiction!

C2 Suppose on the other hand that $\mathbf{M}'_\Pi$ is *not* prime. Then some prime p divides $\mathbf{M}'_\Pi$. (Why?) Now, p itself is either in Π, or not; we hence have two sub-cases. Supposing that p is in Π entails that p divides $\mathbf{M}_\Pi$. But we are operating under the supposition that p divides $\mathbf{M}'_\Pi$ as well. This implies that p divides 1, which is absurd (a contradiction). Hence the prime p is outside Π.

Hence for *any* such list Π, there is a prime outside the list. That is, there are infinitely many primes. **QED**

Euclidean "Magic"

Theorem: There are infinitely many primes.

Proof: We take an indirect route. Let $\Pi = p_1 = 2, p_2 = 3, p_3 = 5, \ldots, p_k$ be a finite, exhaustive consecutive sequence of prime numbers. Next, let $\mathbf{M}_\Pi$ be $p_1 \times p_2 \times \cdots \times p_k$, and set $\mathbf{M}'_\Pi$ to $\mathbf{M}_\Pi + 1$. Either $\mathbf{M}'_\Pi$ is prime, or not; we thus have two (exhaustive) cases to consider.

C1 Suppose $\mathbf{M}'_\Pi$ is prime. In this case we immediately have a prime number beyond any in Π — contradiction!

C2 Suppose on the other hand that $\mathbf{M}'_\Pi$ is *not* prime. Then some prime p divides $\mathbf{M}'_\Pi$. (Why?) Now, p itself is either in Π, or not; we hence have two sub-cases. Supposing that p is in Π entails that p divides $\mathbf{M}_\Pi$. But we are operating under the supposition that p divides $\mathbf{M}'_\Pi$ as well. This implies that p divides 1, which is absurd (a contradiction). Hence the prime p is outside Π.

Hence for *any* such list Π, there is a prime outside the list. That is, there are infinitely many primes. **QED**

350 BC

Euclid

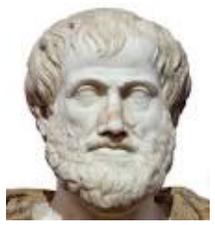
2020

Intro to (Formal) Logic @ RPI

350 BC

Euclid

300 BC

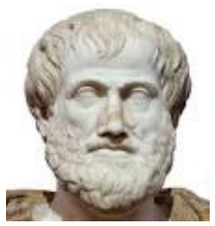


2020

350 BC

300 BC

2020



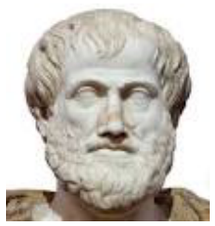
I don't believe in magic! Why exactly is that so convincing? What exactly is he doing?!?

Euclid

350 BC

Euclid

300 BC



Organon

2020

I don't believe in magic! Why exactly is that so convincing? What exactly is he doing?!?

He's using syllogisms!

E.g.,

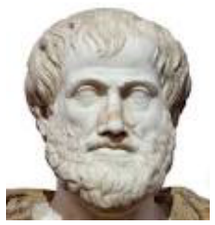
All As are Bs.
All Bs are Cs.

All As are Cs.

350 BC

Euclid

300 BC



Organon

2020

I don't believe in magic! Why exactly is that so convincing? What exactly is he doing?!?

He's using syllogisms!

E.g.,

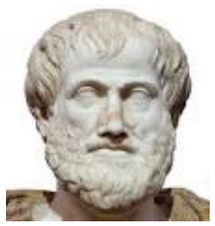
All As are Bs.
All Bs are Cs.

All As are Cs.

350 BC

Euclid

300 BC



Organon

2020

I don't believe in magic! Why exactly is that so convincing? What exactly is he doing?!?

Balderdash!

He's using syllogisms!

E.g.,

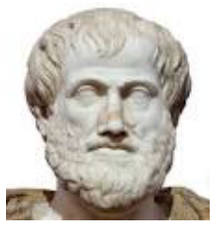
All As are Bs.
All Bs are Cs.

All As are Cs.

350 BC

Euclid

300 BC



Organon

2020

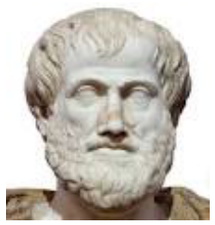
I don't believe in magic! Why exactly is that so convincing? What exactly is he doing?!?

Balderdash!

350 BC

Euclid

300 BC



Organon

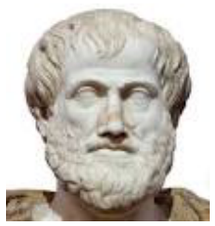
2020

I don't believe in magic! Why exactly is that so convincing? What exactly is he doing?!?

350 BC

Euclid

300 BC

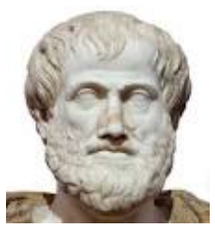


Organon

350 BC

300 BC

1666



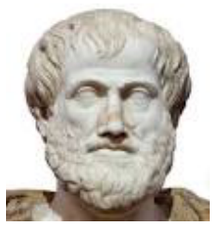
Euclid

Organon

350 BC

Euclid

300 BC



Organon

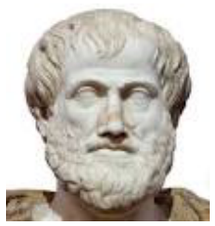
1666

Leibniz

350 BC

Euclid

300 BC



Organon

1666

Leibniz

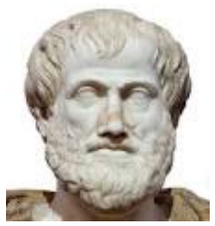
$$\int$$

"Universal Computational Logic"

350 BC

Euclid

300 BC



Organon

1666

Leibniz

$$\int$$

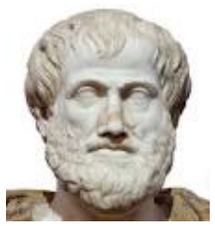
Intro to (Formal) Logic @ RPI

“Universal Computational Logic”

350 BC

Euclid

300 BC



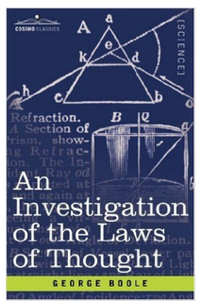
Organon

1666

Leibniz

$\int$

1854



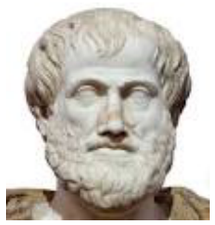
Intro to (Formal) Logic @ RPI

"Universal Computational Logic"

350 BC

Euclid

300 BC



Organon

1666

Leibniz

$$\int$$

1854

Intro to (Formal) Logic @ RPI

“Universal Computational Logic”

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC

Organon

1666

Leibniz

$$\int$$

1854

1956

Simon

Intro to (Formal) Logic @ RPI

“Astonishing” Logic Theorist Proof @ Dawn of AI

“Astonishing” Logic Theorist Proof @ Dawn of AI

1	$(\phi \vee \phi) \rightarrow \phi$	axiom
2	$(\neg\phi \vee \neg\phi) \rightarrow \neg\phi$	substitution
3	$(\phi \rightarrow \neg\phi) \rightarrow \neg\phi$	a “replacement rule”
4	$(A \rightarrow \neg A) \rightarrow \neg A$	substitution

“Astonishing” Logic Theorist Proof @ Dawn of AI

1	$(\phi \vee \phi) \rightarrow \phi$	axiom
2	$(\neg\phi \vee \neg\phi) \rightarrow \neg\phi$	substitution
3	$(\phi \rightarrow \neg\phi) \rightarrow \neg\phi$	a “replacement rule”
4	$(A \rightarrow \neg A) \rightarrow \neg A$	substitution

At dawn of AI: 10 seconds.

"Astonishing" Logic Theorist Proof @ Dawn of AI

1	$(\phi \vee \phi) \rightarrow \phi$	axiom
2	$(\neg\phi \vee \neg\phi) \rightarrow \neg\phi$	substitution
3	$(\phi \rightarrow \neg\phi) \rightarrow \neg\phi$	a "replacement rule"
4	$(A \rightarrow \neg A) \rightarrow \neg A$	substitution

At dawn of AI: 10 seconds.

AI of today, e.g. PC provability oracle in HyperSlate®, *vanishingly* small amount of time.

“Universal Computational Logic”

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

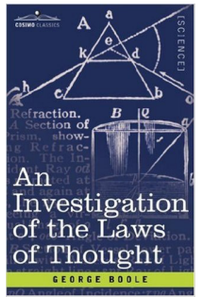
300 BC

Organon

Leibniz

$\int$

1854



1956

Simon

2020

Intro to (Formal) Logic @ RPI

“Universal Computational Logic”

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC

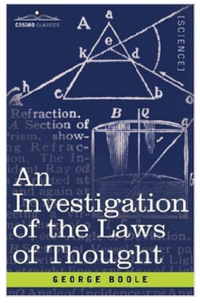
Organon

1666

Leibniz

$\int$

1854



1956

Simon

2020

Intro to (Formal) Logic @ RPI

“Universal Computational Logic”

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC

Organon

1666

Leibniz

$$\int$$

1854

1956

Simon

2020

Intro to (Formal) Logic @ RPI

“Universal Computational Logic”

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC

Organon

1666

Leibniz

$$\int$$

1854

1956

Simon

2020

2021

Intro to (Formal) Logic @ RPI

“Universal Computational Logic”

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC

Organon

1666

Leibniz

$$\int$$

1854

1956

Simon

2020

Intro to (Formal) Logic @ RPI

2021

Thesingularity?

Entscheidungsproblem

"Universal Computational Logic"

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC

Organon

1666

Leibniz

$$\int$$

1854

1956

Simon

2020

Intro to (Formal) Logic @ RPI

2021

Thesingularity?

?

Entscheidungsproblem

"Universal Computational Logic"

Logic Theorist
(birth of modern logicist AI)

The Singularity?

350 BC

Euclid

300 BC

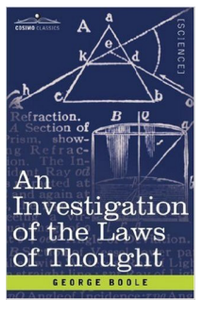
Organon

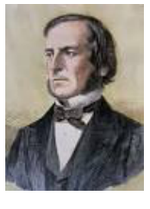
1666

Leibniz

$\int$

1854





1956

Simon

2020

Intro to Logic @ RPI

2021

?

Entscheidungsproblem

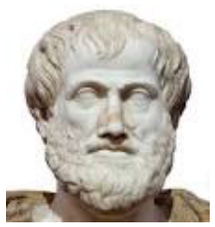
"Universal Computational Logic"

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC



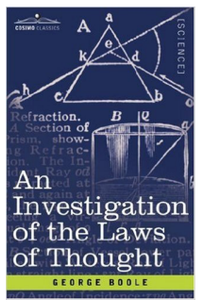
Organon

1666

Leibniz

$\int$

1854



1956

Simon

2020

2021

Intro to Logic @ RPI

Thesingularity?

?

Entscheidungsproblem

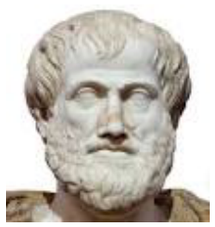
"Universal Computational Logic"

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC



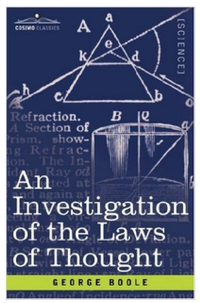
Organon

1666

Leibniz

$\int$

1854



1956

Simon

2020

2021

Frege

Intro to Logic @ RPI

Thesingularity?

?

Entscheidungsproblem

"Universal Computational Logic"

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC

Organon

1666

Leibniz

$\int$

1854

1956

Simon

2020

2021

Intro to Logic @ RPI

Frege

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

Thesingularity?

?

Entscheidungsproblem

"Universal Computational Logic"

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC

Organon

1666

Leibniz

$\int$

1854

1956

Simon

2020

2021

Intro to Logic @ RPI

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

Thesingularity?

?

Entscheidungsproblem

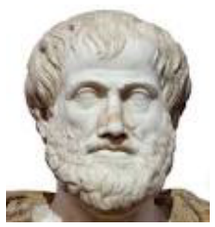
"Universal Computational Logic"

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC



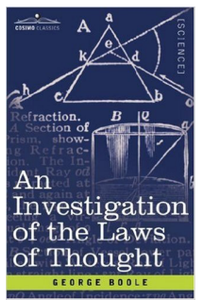
Organon

1666

Leibniz

$\int$

1854



1956

Simon

1970

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

Intro to Logic @ RPI

2021

Thesingularity?

?

Entscheidungsproblem

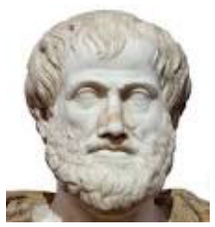
"Universal Computational Logic"

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC



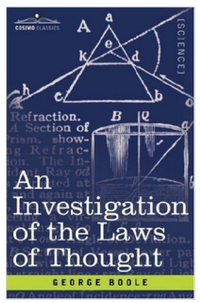
Organon

1666

Leibniz

$\int$

1854



1956

Simon

2020

2021

Intro to Logic @ RPI

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

Thesingularity?

?

Entscheidungsproblem

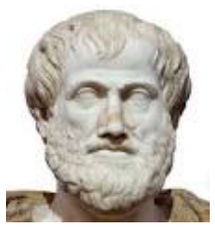
"Universal Computational Logic"

Logic Theorist
(birth of modern logicist AI)

350 BC

Euclid

300 BC



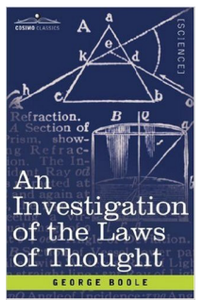
Organon

1666

Leibniz

$\int$

1854



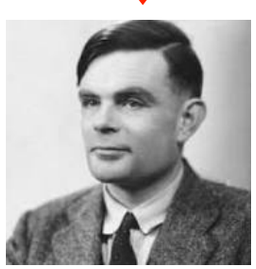
1956

Simon

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 Logic @ RPI

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

2020

2021

Thesingularity?

First, the Theoremhood Decision Problem (THEOREM$_{PC}$) for the Propositional Calculus

First, the Theoremhood Decision Problem (THEOREM$_{PC}$) for the Propositional Calculus

First, the Theoremhood Decision Problem (THEOREM_{PC}) for the Propositional Calculus

First, the Theoremhood Decision Problem (THEOREM$_{PC}$) for the Propositional Calculus

First, the Theoremhood Decision Problem (THEOREM$_{PC}$) for the Propositional Calculus

First, the Theoremhood Decision Problem (THEOREM$_{PC}$) for the Propositional Calculus

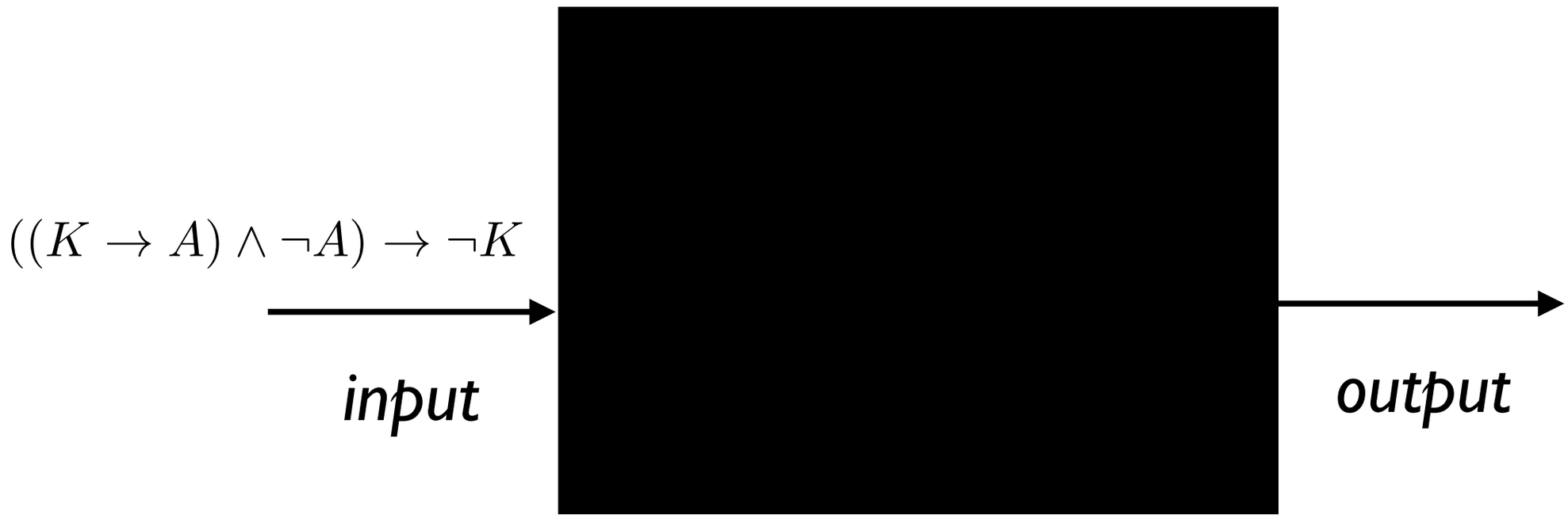
First, the Theoremhood Decision Problem (THEOREM$_{PC}$) for the Propositional Calculus

Hard!! — for apparently no polynomial-time algorithm for this!

First, the Theoremhood Decision Problem (THEOREM$_{PC}$) for the Propositional Calculus

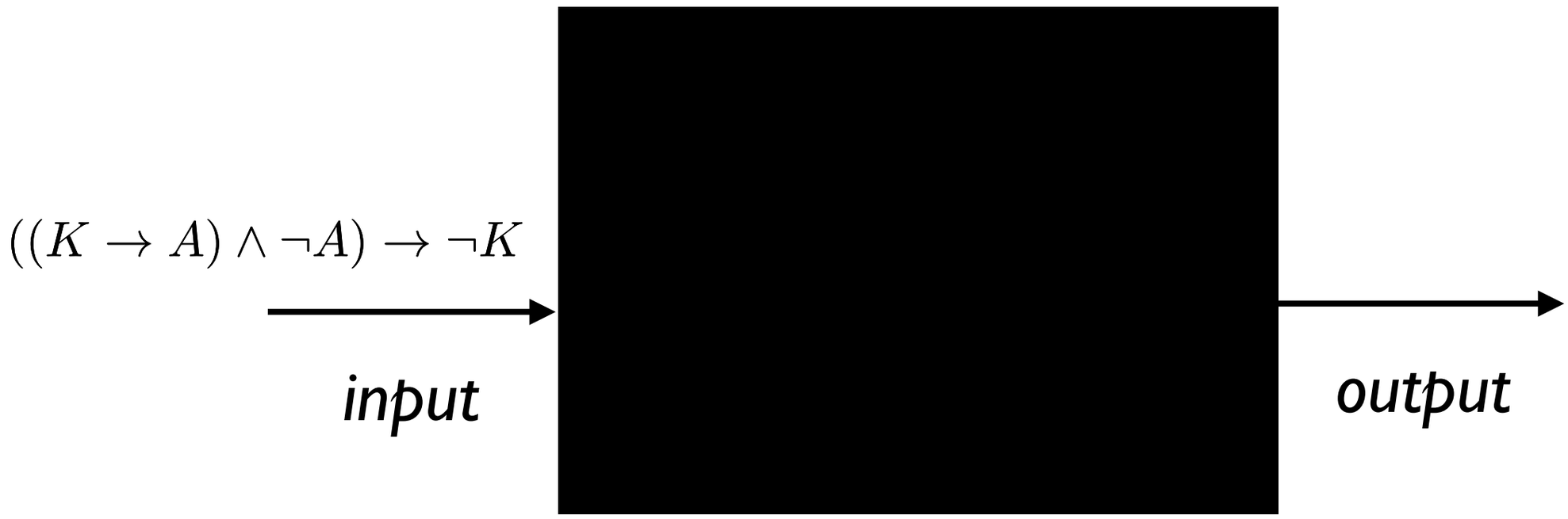
Hard!! — for apparently no polynomial-time algorithm for this!

First, the Theoremhood Decision Problem ($\text{THEOREM}_{\text{PC}}$) for the Propositional Calculus



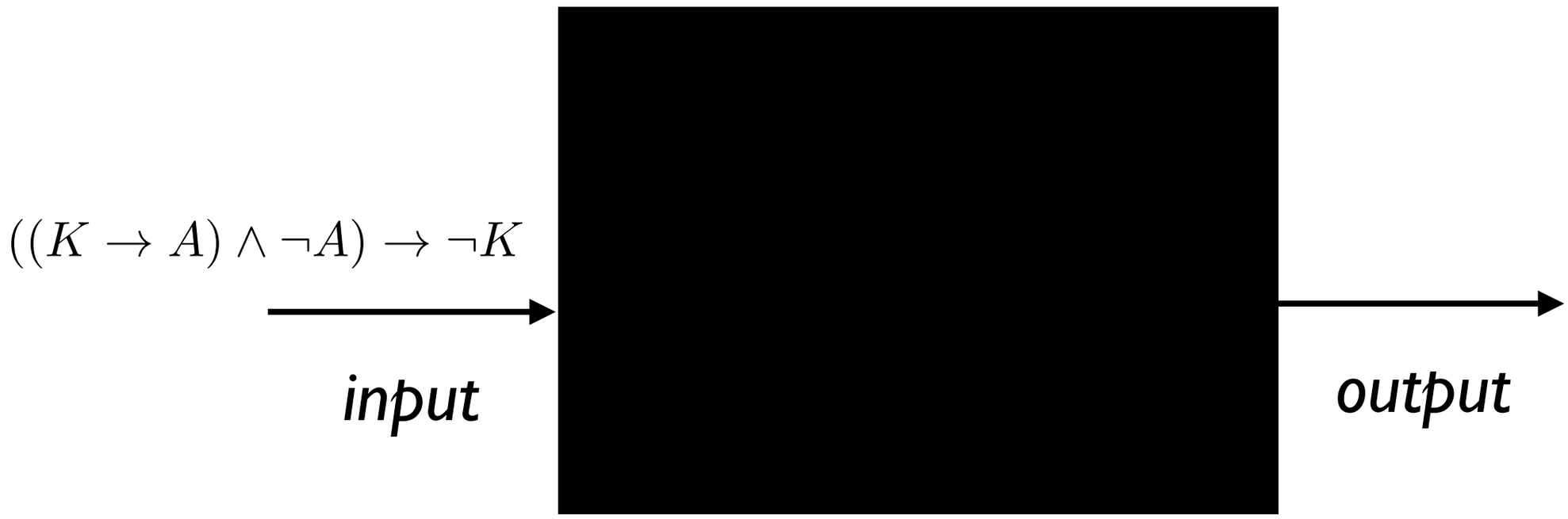
Hard!! — for apparently no polynomial-time algorithm for this!

First, the Theoremhood Decision Problem (THEOREM_{PC}) for the Propositional Calculus



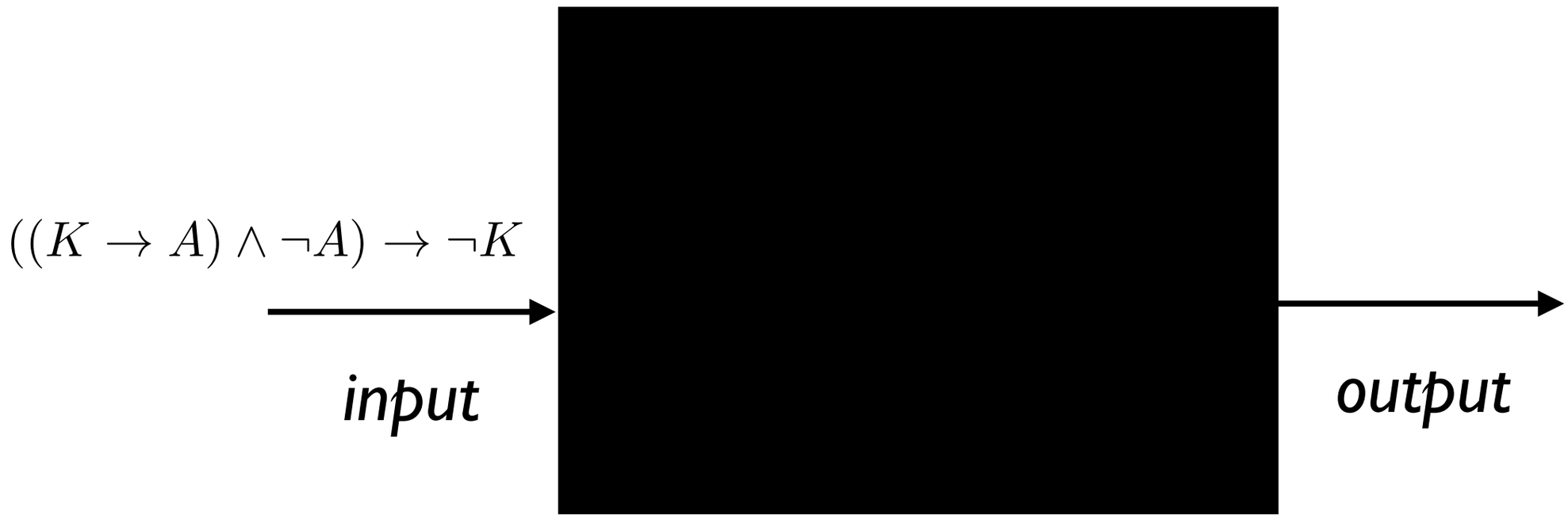
Hard!! — for apparently no polynomial-time algorithm for this!

First, the Theoremhood Decision Problem ($\text{THEOREM}_{\text{PC}}$) for the Propositional Calculus



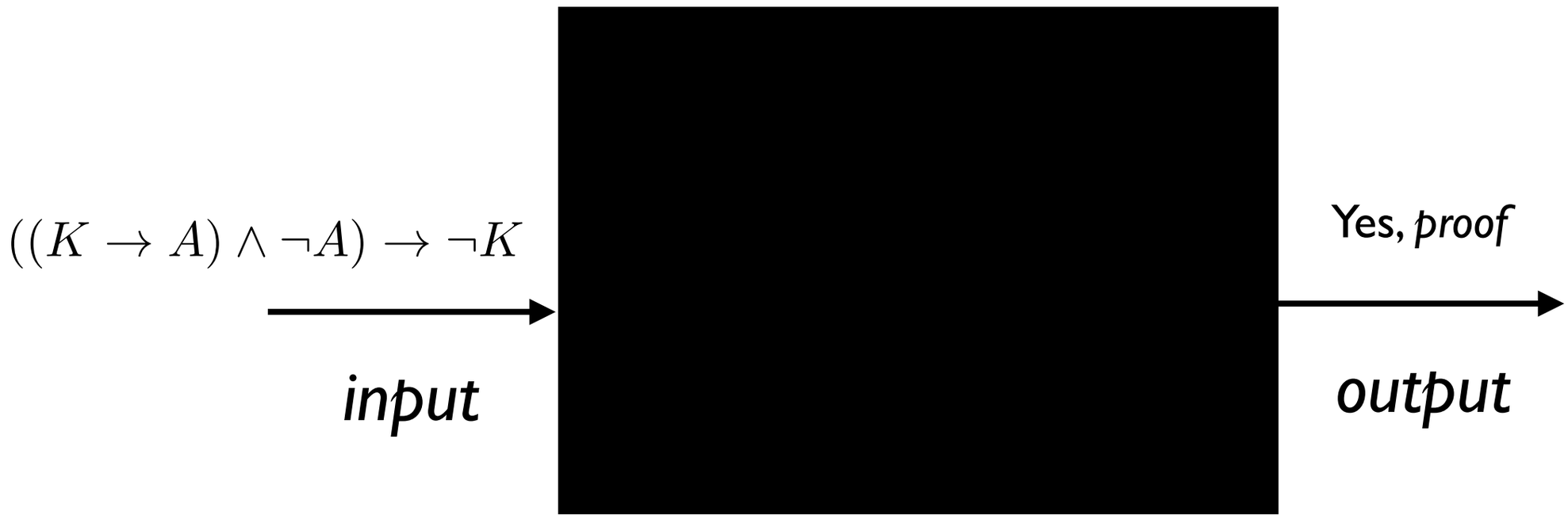
Hard!! — for apparently no polynomial-time algorithm for this!

First, the Theoremhood Decision Problem ($\text{THEOREM}_{\text{PC}}$) for the Propositional Calculus



Hard!! — for apparently no polynomial-time algorithm for this!

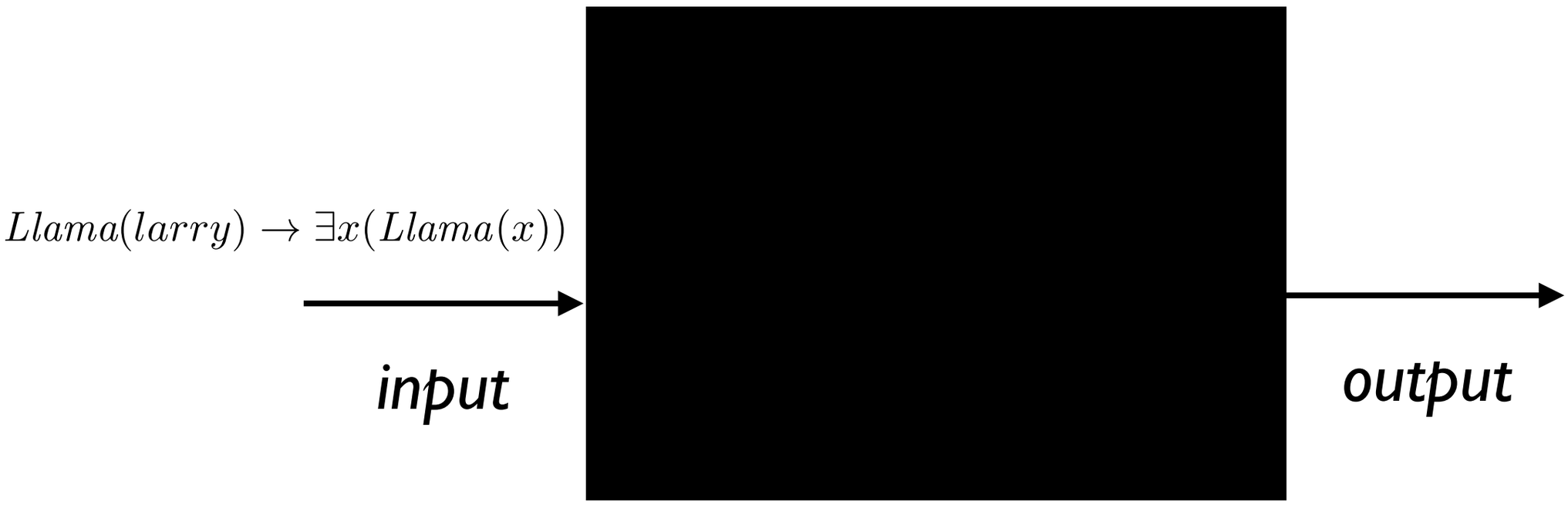
First, the Theoremhood Decision Problem ($\text{THEOREM}_{\text{PC}}$) for the Propositional Calculus



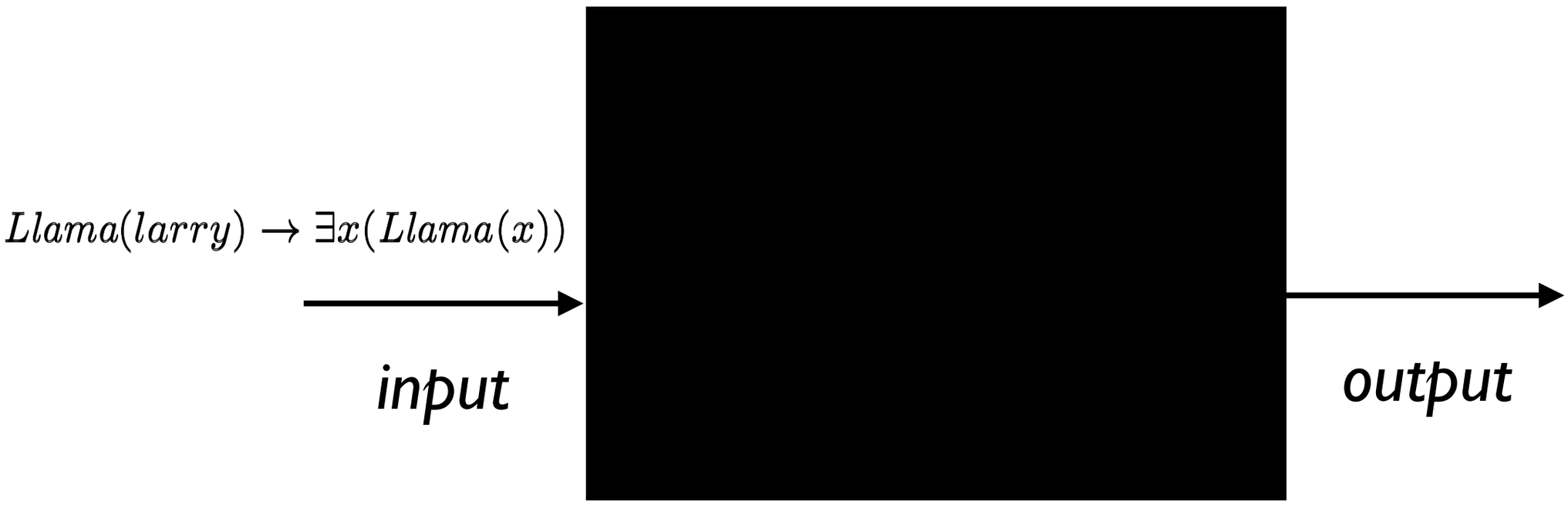
Hard!! — for apparently no polynomial-time algorithm for this!

And now, the Theoremhood Decision Problem,
i.e., the *Entscheidungsproblem*,
($\text{THEOREM}_{\text{FOL}}$)
for First-Order Logic (FOL)

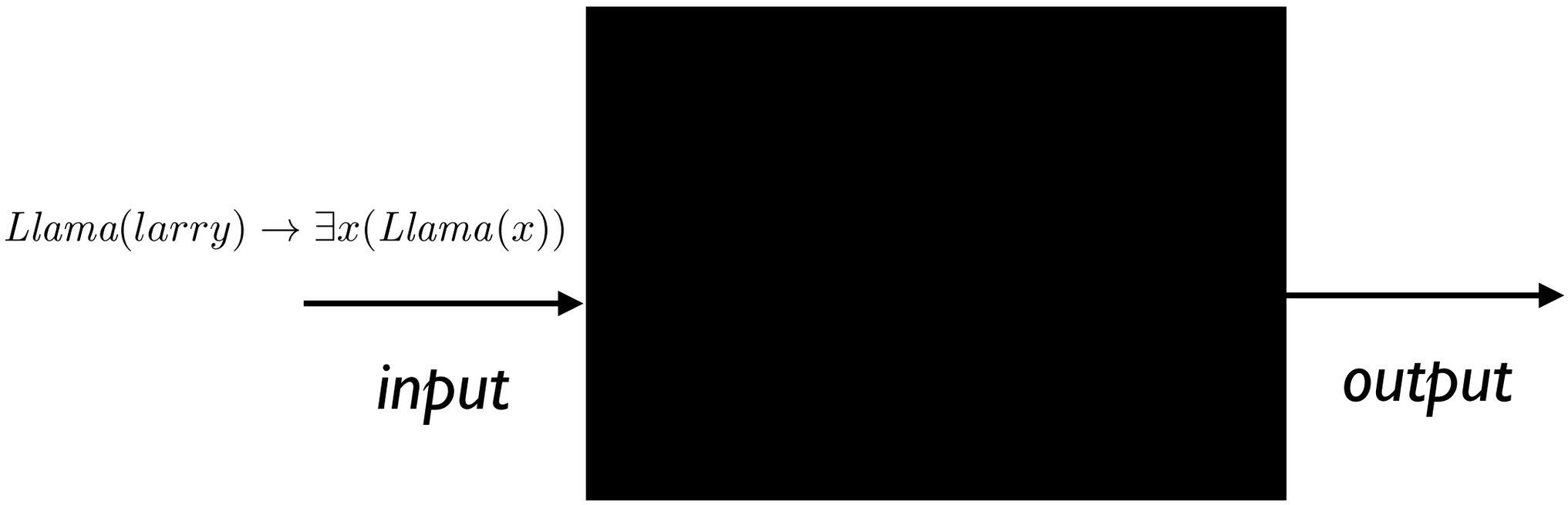
And now, the Theoremhood Decision Problem,
i.e., the *Entscheidungsproblem*,
($\text{THEOREM}_{\text{FOL}}$)
for First-Order Logic (FOL)



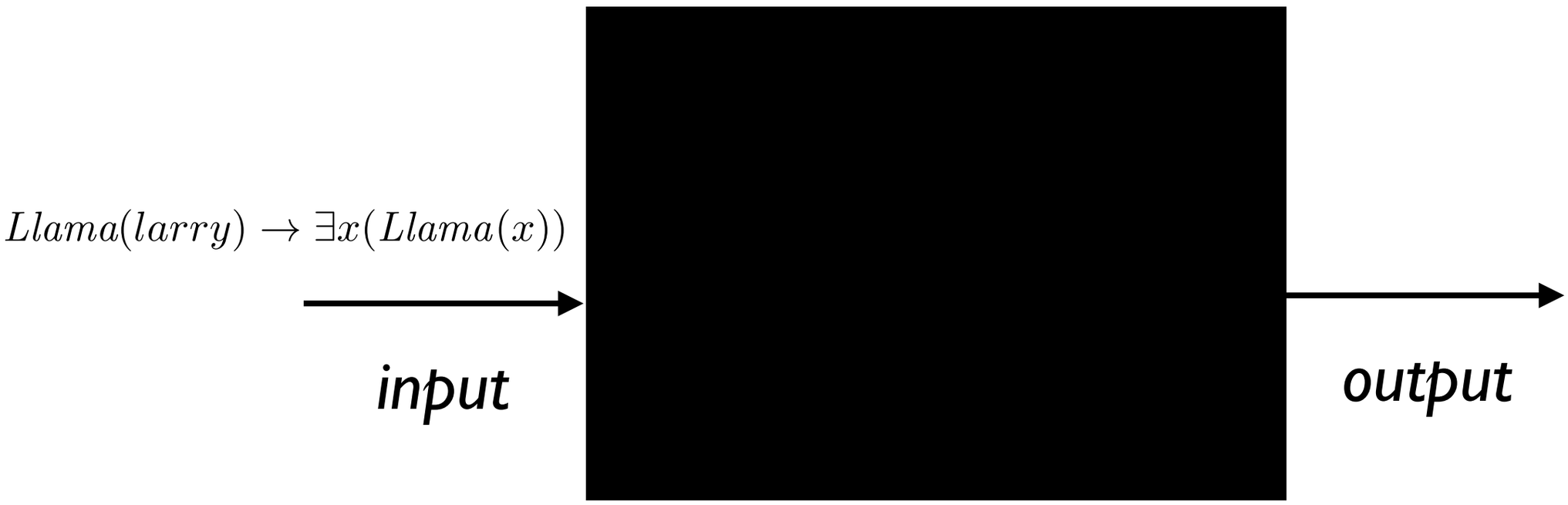
And now, the Theoremhood Decision Problem, i.e., the *Entscheidungsproblem*, ($\text{THEOREM}_{\text{FOL}}$) for First-Order Logic (FOL)



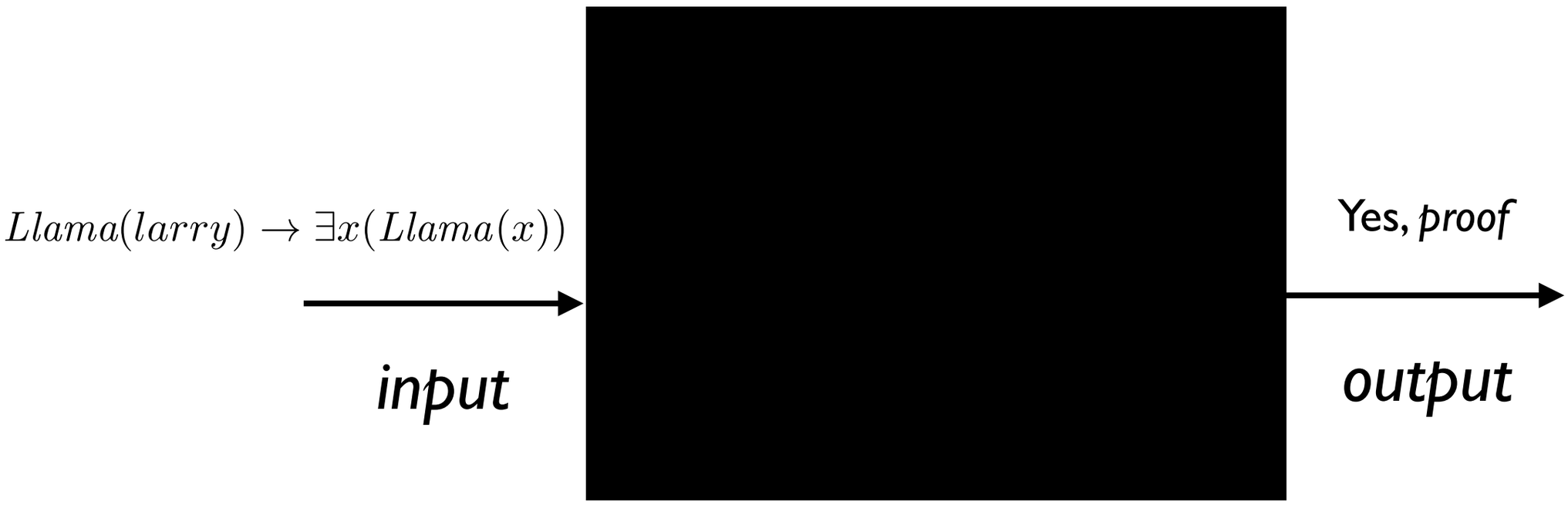
And now, the Theoremhood Decision Problem,
i.e., the *Entscheidungsproblem*,
($\text{THEOREM}_{\text{FOL}}$)
for First-Order Logic (FOL)



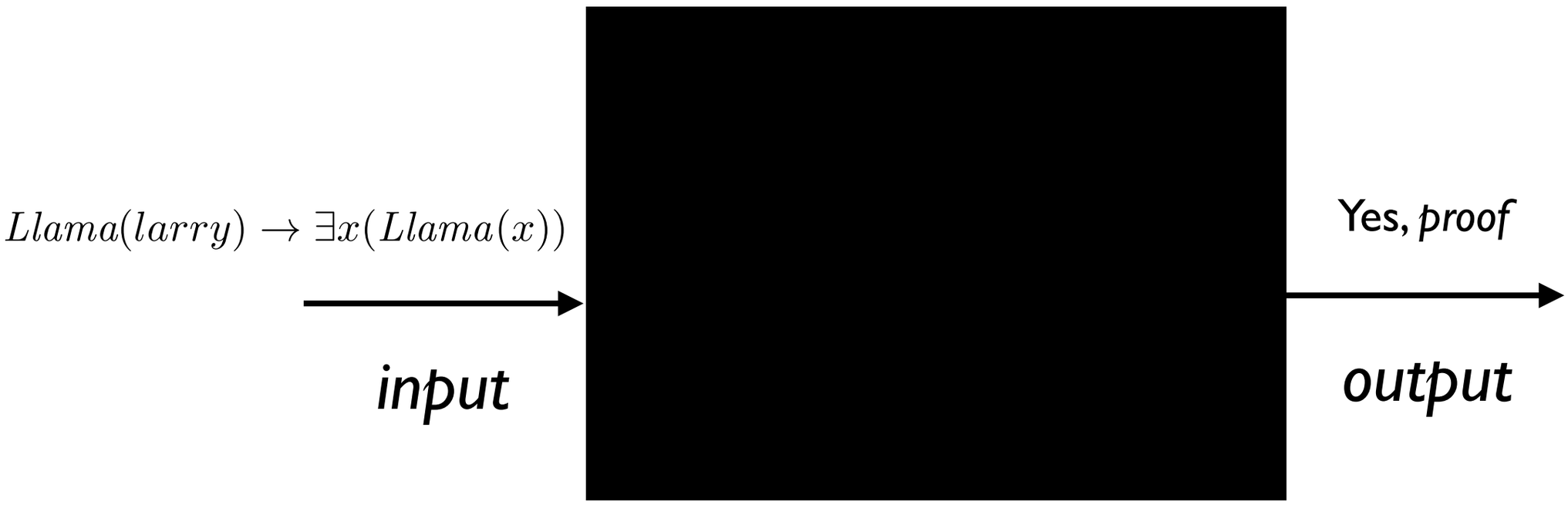
And now, the Theoremhood Decision Problem,
i.e., the *Entscheidungsproblem*,
($\text{THEOREM}_{\text{FOL}}$)
for First-Order Logic (FOL)



And now, the Theoremhood Decision Problem, i.e., the *Entscheidungsproblem*, ($\text{THEOREM}_{\text{FOL}}$) for First-Order Logic (FOL)



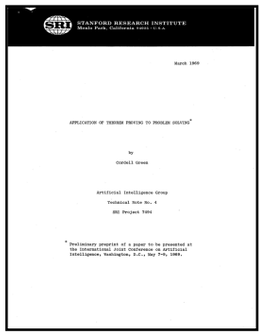
And now, the Theoremhood Decision Problem, i.e., the *Entscheidungsproblem*, ($\text{THEOREM}_{\text{FOL}}$) for First-Order Logic (FOL)



Not just hard: *impossible* for a (and this needed to be *invented* in the course of clarifying and solving the problem) standard computing machine.

On Logic & AI,

Specifically …



STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 U.S.A.

March 1969

APPLICATION OF THEOREM PROVING TO PROBLEM SOLVING*

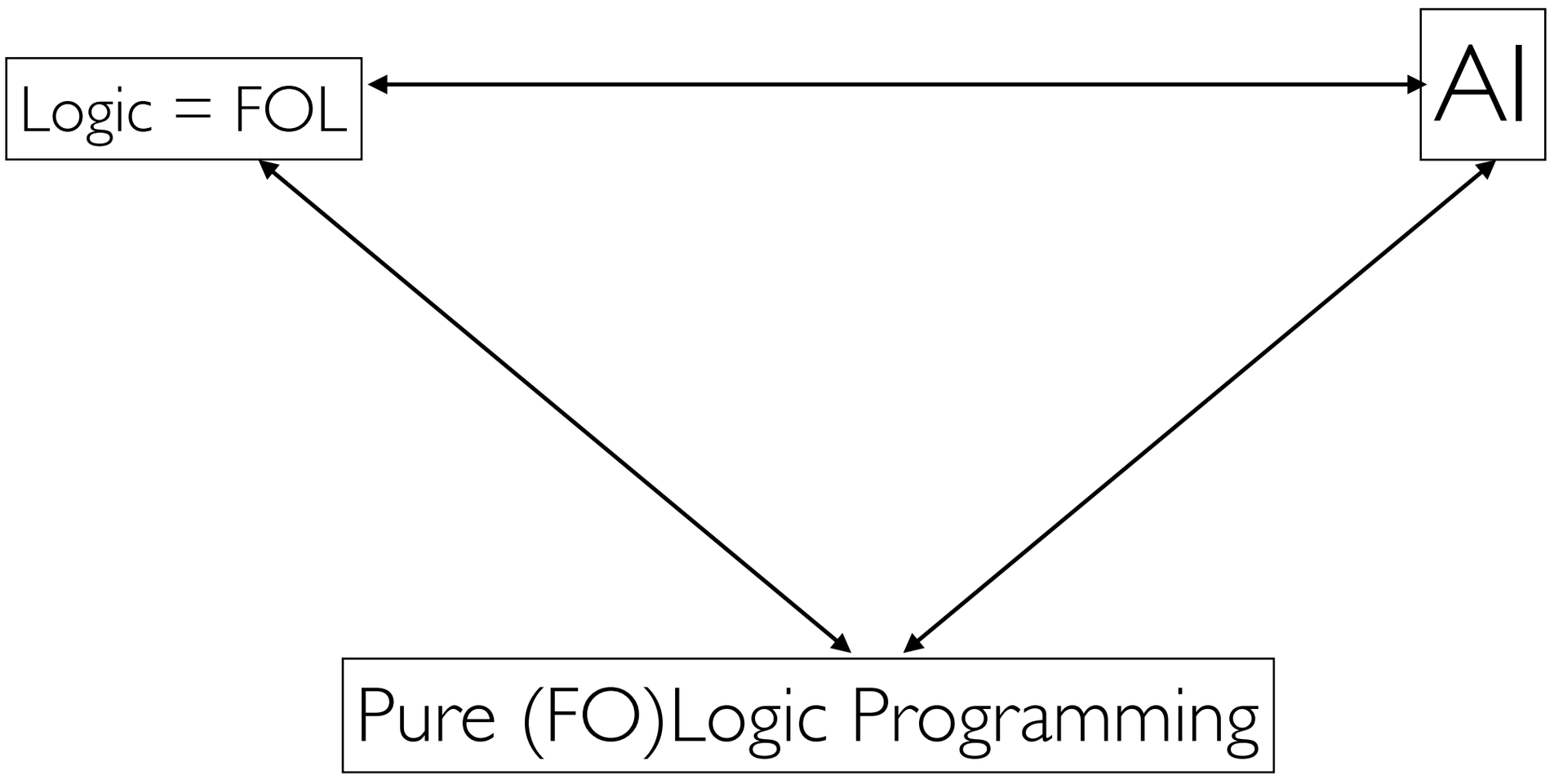
by

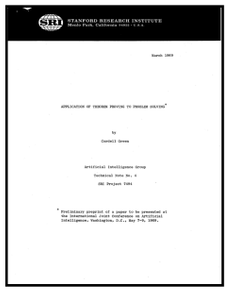
Cordell Green

Artificial Intelligence Group
Technical Note No. 4
SRI Project 7494

* Preliminary preprint of a paper to be presented at the International Joint Conference on Artificial Intelligence, Washington, D.C., May 7-9, 1969.

The Terrific Triad circa 1965





STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025

March 1969

APPLICATION OF THEOREM PROVING TO PROBLEM SOLVING*

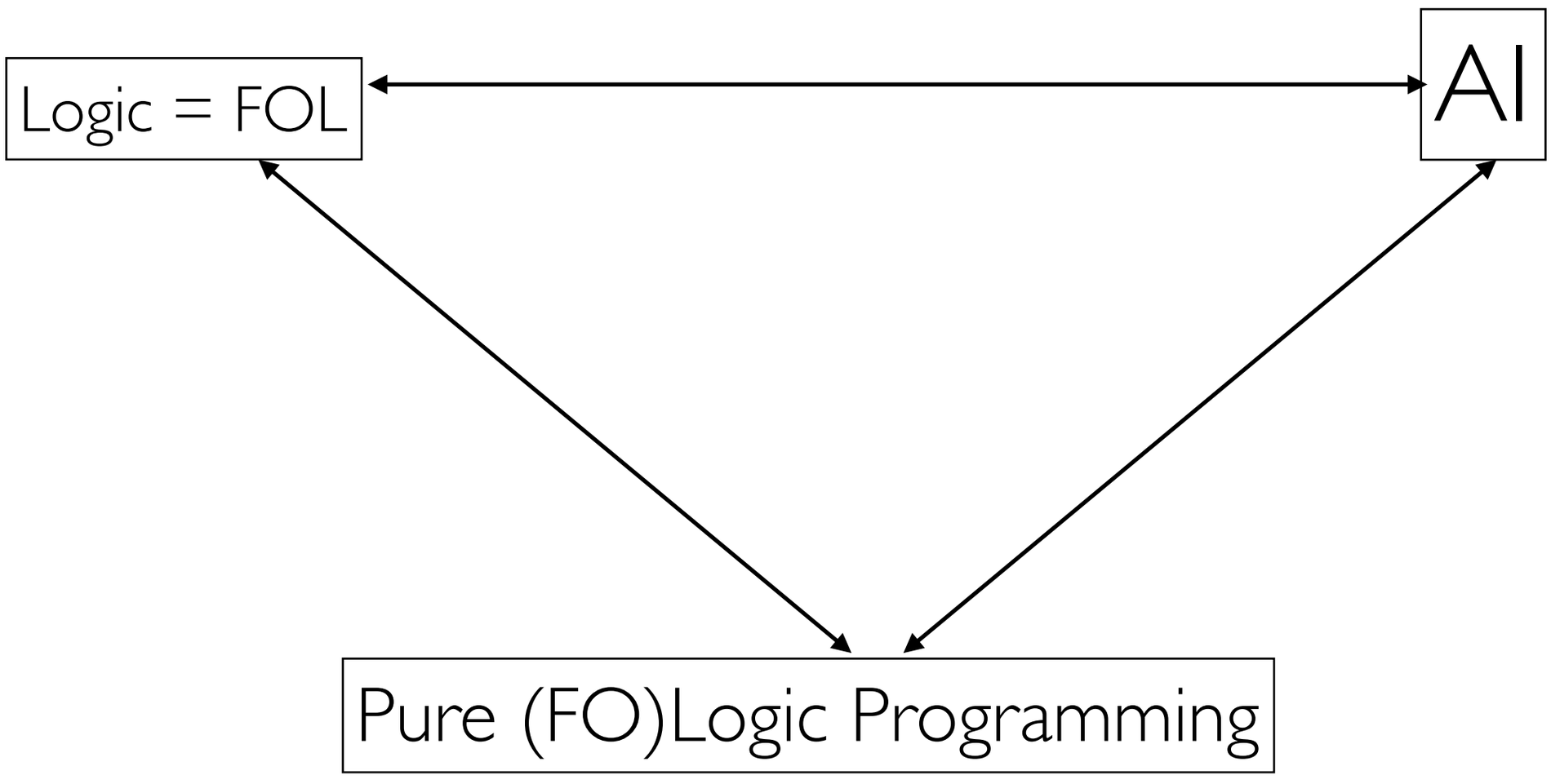
by

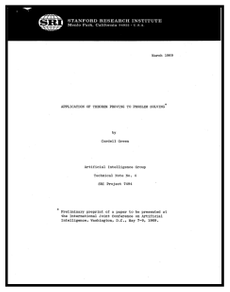
Cordell Green

Artificial Intelligence Group
Technical Note No. 6
SRI Project 7184

* Preliminary preprint of a paper to be presented at the International Joint Conference on Artificial Intelligence, Washington, D.C., May 7-9, 1969.

The Disastrous Disunion





STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025

March 1969

APPLICATION OF THEOREM PROVING TO PROBLEM SOLVING*

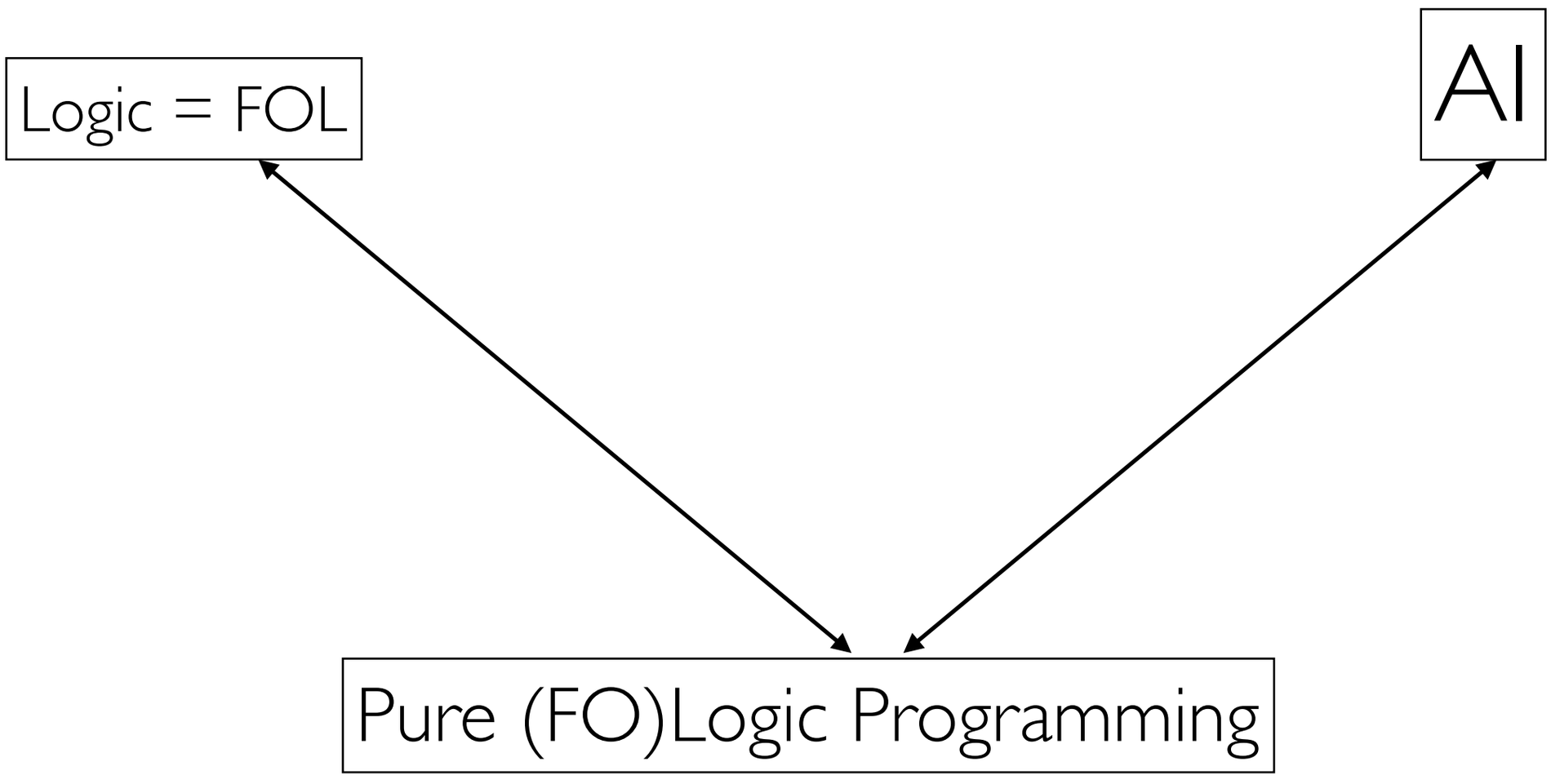
by

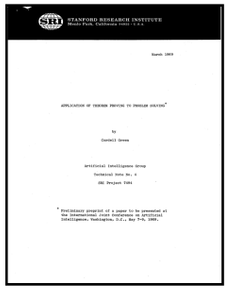
Cordell Green

Artificial Intelligence Group
Technical Note No. 6
SRI Project 7184

* Preliminary preprint of a paper to be presented at the International Joint Conference on Artificial Intelligence, Washington, D.C., May 7-9, 1969.

The Disastrous Disunion





STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025

March 1969

APPLICATION OF THEOREM PROVING TO PROBLEM SOLVING*

by

Cordell Green

Artificial Intelligence Group
Technical Note No. 6
SRI Project 7184

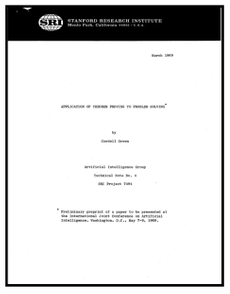
* Preliminary preprint of a paper to be presented at the International Joint Conference on Artificial Intelligence, Washington, D.C., May 7-9, 1969.

The Disastrous Disunion

Logic = FOL

AI

Pure (FO)Logic Programming


STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025

March 1969

APPLICATION OF THEOREM PROVING TO PROBLEM SOLVING*

by

Cordell Green

Artificial Intelligence Group
Technical Note No. 6
SRI Project 7184

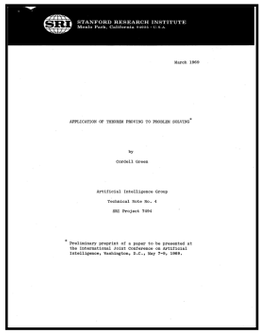
* Preliminary preprint of a paper to be presented at the International Joint Conference on Artificial Intelligence, Washington, D.C., May 7-9, 1969.

The Disastrous Disunion

Logic = FOL

AI

Pure (FO)Logic Programming



STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 U.S.A.

March 1969

APPLICATION OF THEOREM PROVING TO PROBLEM SOLVING*

by

Cordell Green

Artificial Intelligence Group
Technical Note No. 4
SRI Project 7494

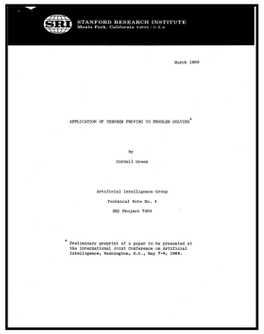
* Preliminary preprint of a paper to be presented at the International Joint Conference on Artificial Intelligence, Washington, D.C., May 7-9, 1969.

The Disastrous Disunion

AI

Logic = FOL

Pure (FO)Logic Programming



STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 U.S.A.

March 1969

APPLICATION OF THEOREM PROVING TO PROBLEM SOLVING*

by

Cordell Green

Artificial Intelligence Group
Technical Note No. 4
SRI Project 7494

* Preliminary preprint of a paper to be presented at the International Joint Conference on Artificial Intelligence, Washington, D.C., May 7-9, 1969.

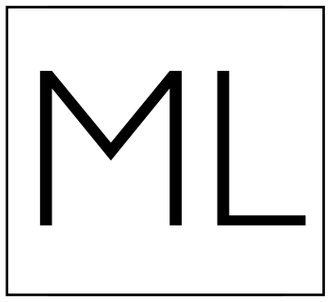
The Disastrous Disunion

ML

Logic = FOL

Pure (FO)Logic Programming

The Disastrous Disunion



Logic = FOL

Pure (FO)Logic Programming

Some Disastrous Consequences …

Disastrous Consequence #1:
Mindless Procedure is Venerated & Pushed

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

ingrained in everyone's lives when words like algorithm and precondition are part of everyone's vocab-

Computational thinking thus has the following characteristics:

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incentuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

Teach computer programming!
(**procedural**, o-o, functional)

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

Teach computer programming!
(**procedural**, o-o, functional)

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incentuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

Teach computer programming!
(**procedural**, o-o, functional)

Computer science is the scientific (or STEM) study of:

what problems can be solved,
what tasks can be accomplished,
and what features of the world can be understood …

… *computationally*, that is, using a language with only:

2 nouns (‘0’, ‘1’),
3 verbs (‘move’, ‘print’, ‘halt’),
3 grammar rules (sequence, selection, repetition),
and nothing else,

and then to provide algorithms to show how this can be done:

efficiently,
practically,
physically,
and ethically.

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incentuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

→ Teach computer programming! (**procedural**, o-o, functional) →

Computer science is the scientific (or STEM) study of:

what problems can be solved,
what tasks can be accomplished,
and what features of the world can be understood …

… *computationally*, that is, using a language with only:

2 nouns (‘0’, ‘1’),
3 verbs (‘move’, ‘print’, ‘halt’),
3 grammar rules (sequence, selection, repetition),
and nothing else,

and then to provide algorithms to show how this can be done:

efficiently,
practically,
physically,
and ethically.

– Rapaport, “phics" book

Disastrous Consequence #2: Impenetrable, Dangerous Code

The Coming Software Apocalypse

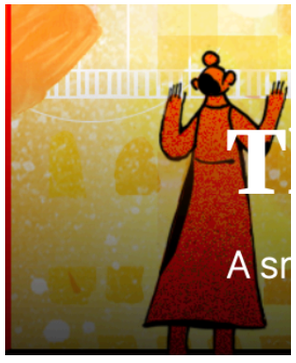
A small group of programmers wants to change how we code—before catastrophe strikes.

In September 2007, Jean Bookout was driving on the highway with her best friend in a Toyota Camry when the accelerator seemed to get stuck. When she took her foot off the pedal, the car didn't slow down. She tried the brakes but they seemed to have lost their power. As she swerved toward an off-ramp going 50 miles per hour, she pulled the emergency brake. The car left a skid mark 150 feet long before running into an embankment by the side of the road. The passenger was killed. Bookout woke up in a hospital a month later.

The incident was one of many in a nearly decade-long investigation into claims of so-called unintended acceleration in Toyota cars. Toyota blamed the incidents on poorly designed floor mats, "sticky" pedals, and driver error, but outsiders suspected that faulty software might be responsible. The National Highway Traffic Safety Administration enlisted software experts from NASA to perform an intensive review of Toyota's code. After nearly 10 months, the NASA team hadn't found evidence that software was the cause—but said they couldn't prove it wasn't.

It was during litigation of the Bookout accident that someone finally found a convincing connection. Michael Barr, an expert witness for the plaintiff, had a team of software experts spend 18 months with the Toyota code, picking up where NASA left off. Barr described what they found as "spaghetti code," programmer lingo for software that has become a tangled mess. Code turns to spaghetti when it accretes over many years, with feature after feature piling on top of, and being woven around, what's already there; eventually the code becomes impossible to follow, let alone to test exhaustively for flaws.

mark 150 feet long before running into an embankment by the side of the road. The passenger was killed. Bookout woke up in a hospital a month later.



The incident was one of many in a nearly decade-long investigation into claims of so-called unintended acceleration in Toyota cars. Toyota blamed the incidents on poorly designed floor mats, “sticky” pedals, and driver error, but outsiders suspected that faulty software might be responsible. The National Highway Traffic Safety Administration enlisted software experts from NASA to perform an intensive review of Toyota’s code. After nearly 10 months, the NASA team hadn’t found evidence that software was the cause—but said they couldn’t prove it wasn’t.

It was during litigation of the Bookout accident that someone finally found a convincing connection. Michael Barr, an expert witness for the plaintiff, had a team of software experts spend 18 months with the Toyota code, picking up where NASA left off. Barr described what they found as “spaghetti code,” programmer lingo for software that has become a tangled mess. Code turns to spaghetti when it accretes over many years, with feature after feature piling on top of, and being woven around, what’s already there; eventually the code becomes impossible to follow, let alone to test exhaustively for flaws.

Disastrous Consequence #3: Black-Box Machine-Learning Machines that Don't Learn Anything At All

Since Plato:

Knowledge is justified, true belief — where justifications (arguments and proofs) are necessarily based on logic.

But Plato has been trampled.

Do Machine-Learning Machines Learn?

Selmer Bringsjord and Naveen Sundar Govindarajulu and Shreya Banerjee and John Hummel

Abstract We answer the present paper's title in the negative. We begin by introducing and characterizing "real learning" ($\mathcal{RL}$) in the formal sciences, a phenomenon that has been firmly in place in homes and schools since at least Euclid. The defense of our negative answer pivots on an integration of *reductio* and proof by cases, and constitutes a general method for showing that any contemporary form of machine learning (ML) isn't real learning. Along the way, we canvass the many different conceptions of "learning" in not only AI, but psychology and its allied disciplines; none of these conceptions (with one exception arising from the view of cognitive development espoused by Piaget), aligns with real learning. We explain in this context by four steps how to broadly characterize and arrive at a focus on $\mathcal{RL}$.

Selmer Bringsjord
Rensselaer Polytechnic Institute, 110 8th Street Troy, NY USA 12180, e-mail: selmerbringsjord@gmail.com

Naveen Sundar Govindarajulu
Rensselaer Polytechnic Institute, 110 8th Street Troy, NY USA 12180, e-mail: Naveen.Sundar.G@gmail.com

Shreya Banerjee
Rensselaer Polytechnic Institute, 110 8th Street Troy, NY USA 12180, e-mail: shreyabbanerjee@gmail.com

John Hummel
901 West Illinois Street, Urbana, IL 61801, e-mail: jehummel@illinois.edu

8 Appendix: The Formal Method

The following deduction uses fonts in an obvious and standard way to sort between functions ($\mathfrak{f}$), agents ($\mathfrak{a}$), and computing machines ($\mathfrak{m}$) in the Arithmetical Hierarchy. Ordinary italicized Roman is used for particulars under these sorts (e.g. f is a particular function). In addition, '$\mathcal{C}$' denotes any collection of conditions constituting jointly necessary-and-sufficient conditions for a form of current ML, which can come from relevant textbooks (e.g. Luger, 2008; Russell and Norvig, 2009) or papers; we leave this quite up to the reader, as no effect upon the validity of the deductive inference chain will be produced by the preferred instantiation of '$\mathcal{C}$.' It will perhaps be helpful to the reader to point out that the deduction eventuates in the proposition that no machine in the ML fold that in this style learns a relevant function $\mathfrak{f}$ thereby also real-learns $\mathfrak{f}$. We encode this target as follows:

$$(\star)\quad \neg\exists \mathfrak{m}\ \exists \mathfrak{f}\ [\phi := \mathit{MLlearns}(\mathfrak{m},\mathfrak{f}) \wedge \psi := \mathit{RLlearns}(\mathfrak{m},\mathfrak{f}) \wedge \mathcal{C}_{\phi}(\mathfrak{m},\mathfrak{f}) \vdash^{*} (\mathrm{ci}')\text{–}(\mathrm{ciii})_{\psi}(\mathfrak{m},\mathfrak{f})]$$

Note that $(\star)$ employs meta-logical machinery to refer to particular instantiations of $\mathcal{C}$ for a particular, arbitrary case of ML (ϕ is the atomic sub-formula that can be instantiated to make the particular case), and particular instantiations of the triad (ci′)–(ciii) for a particular, arbitrary case of $\mathcal{RL}$ (ψ is the atomic sub-formula that can be instantiated to make the particular case). Meta-logical machinery also allows us to use a provability predicate to formalize the notion that real learning is produced by the relevant instance of ML. If we "pop" ϕ/ψ to yield ϕ'/ψ' we are dealing with the particular instantiation of the atomic sub-formula.

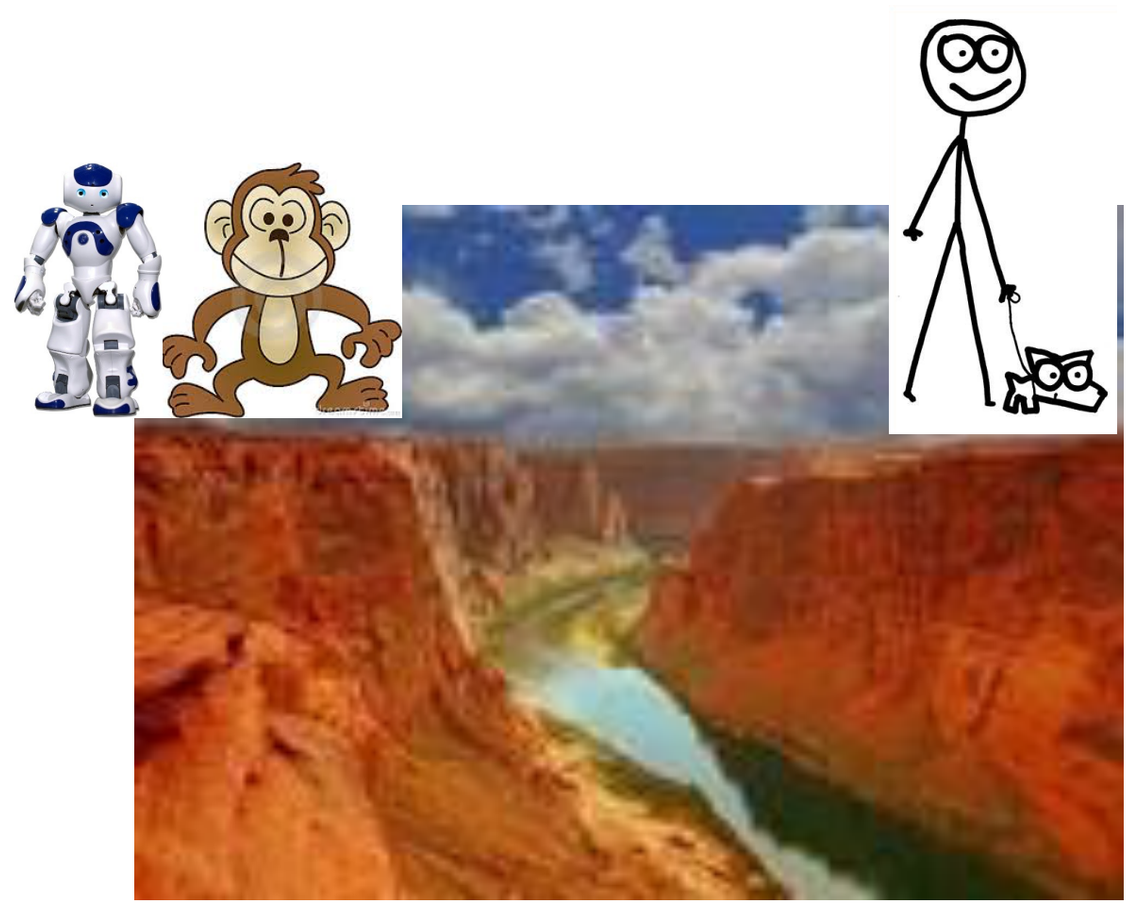
The deduction, as noted in earlier when the informal argument was given, is indirect proof by cases; accordingly, we first assume $\neg(\star)$, and then proceed as follows under this supposition.

	(1)	$\forall \mathfrak{f},\mathfrak{a}\ [\mathfrak{f} : \mathbb{N} \mapsto \mathbb{N} \rightarrow (\mathit{RLlearns}(\mathfrak{a},\mathfrak{f}) \rightarrow \mathrm{(i)}\text{–}\mathrm{(iii)})]$	Def of Real Learning
	(2)	$\mathit{MLlearns}(m,f) \wedge \mathit{RLlearns}(m,f) \wedge f : \mathbb{N} \mapsto \mathbb{N}$	supp (for $\exists$ elim on $(\star)$)
	(3)	$\forall \mathfrak{m},\mathfrak{f}\ [\mathfrak{f} : \mathbb{N} \mapsto \mathbb{N} \rightarrow (\mathit{MLlearns}(\mathfrak{m},\mathfrak{f}) \leftrightarrow \mathcal{C}(\mathfrak{m},\mathfrak{f}))]$	Def of ML
	(4)	$\forall \mathfrak{f}\ [\mathfrak{f} : \mathbb{N} \mapsto \mathbb{N} \rightarrow (\mathit{TurComp}(\mathfrak{f}) \vee \mathit{TurUncomp}(\mathfrak{f}))]$	theorem
	(5)	$\mathit{TurUncomp}(f)$	supp; Case 1
	(6)	$\neg\exists \mathfrak{m}\ \exists \mathfrak{f}\ [(\mathfrak{f} : \mathbb{N} \mapsto \mathbb{N} \wedge \mathit{TurUncomp}(\mathfrak{f}) \wedge \mathcal{C}(\mathfrak{m},\mathfrak{f})]$	theorem
$\therefore$	(7)	$\neg\exists \mathfrak{m}\ \mathit{MLlearns}(\mathfrak{m},f)$	(6), (3)
$\therefore$	(8)	$\bot$	(7), (2)
	(9)	$\mathit{TurComp}(f)$	supp; Case 2
$\therefore$	(10)	$\mathcal{C}_{\phi'}(m,f)$	(2), (3)
$\therefore$	(11)	$(\mathrm{ci}')\text{–}(\mathrm{ciii})_{\psi'}(m,f)$	from supp for $\exists$ elim on $(\star)$ and provability
$\therefore$	(12)	$\neg(\mathrm{ci}')\text{–}(\mathrm{ciii})_{\psi'}(m,f)$	inspection: proofs wholly absent from $\mathcal{C}$
$\therefore$	(13)	$\bot$	(11), (12)
$\therefore$	(14)	$\bot$	*reductio*; proof by cases

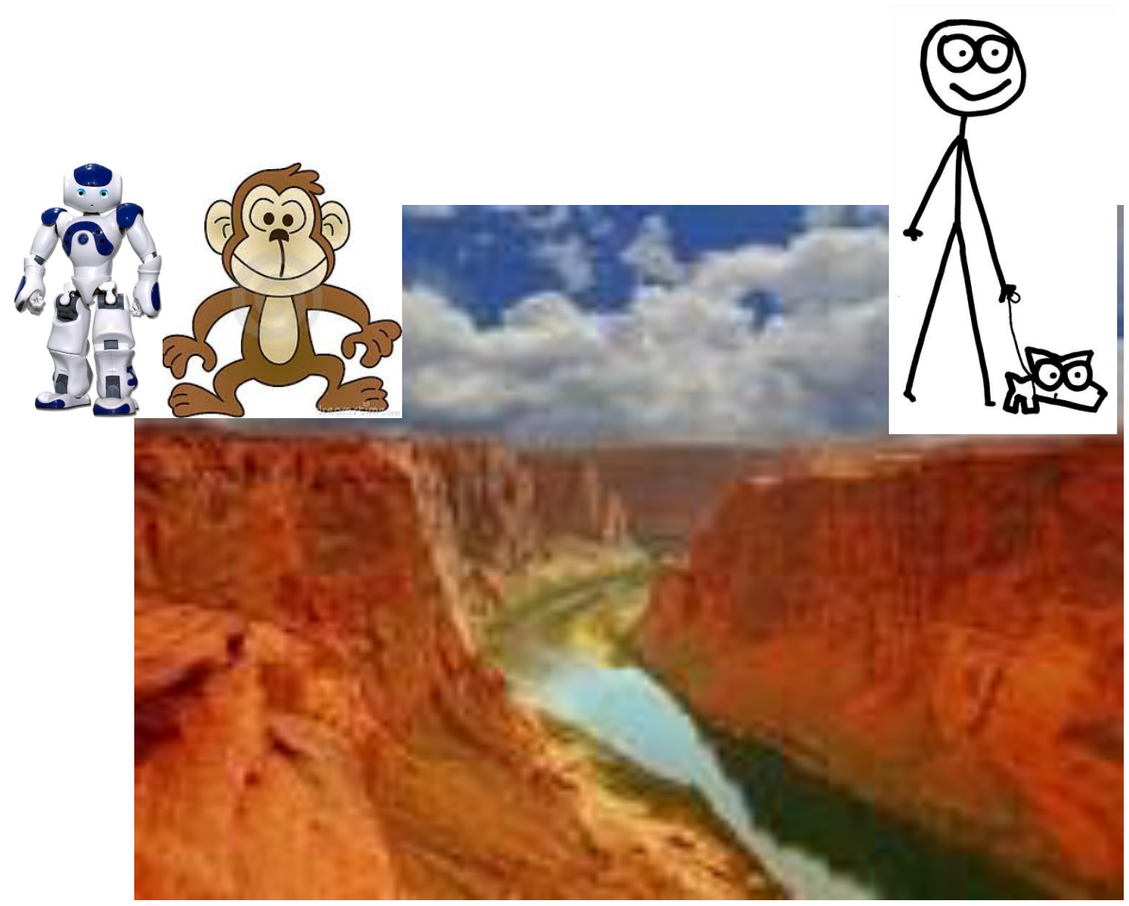
http://kryten.mm.rpi.edu/SB_NSG_SB_JH_DoMachine-LearningMachinesLearn_preprint.pdf

Disastrous Consequence #4: Animal-level AI, Let Alone AI Not Chained to Earth

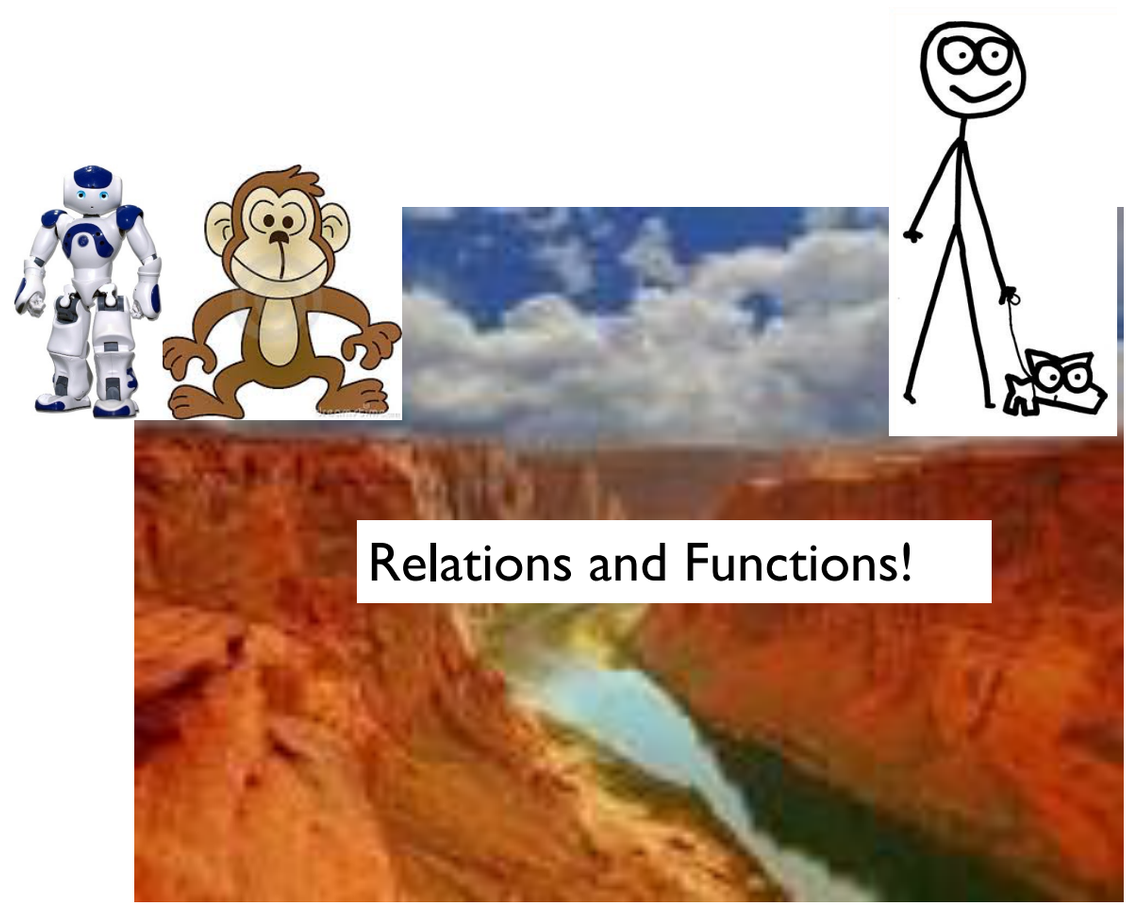
The Canyon of Discontinuity (or Darwin's Dread)



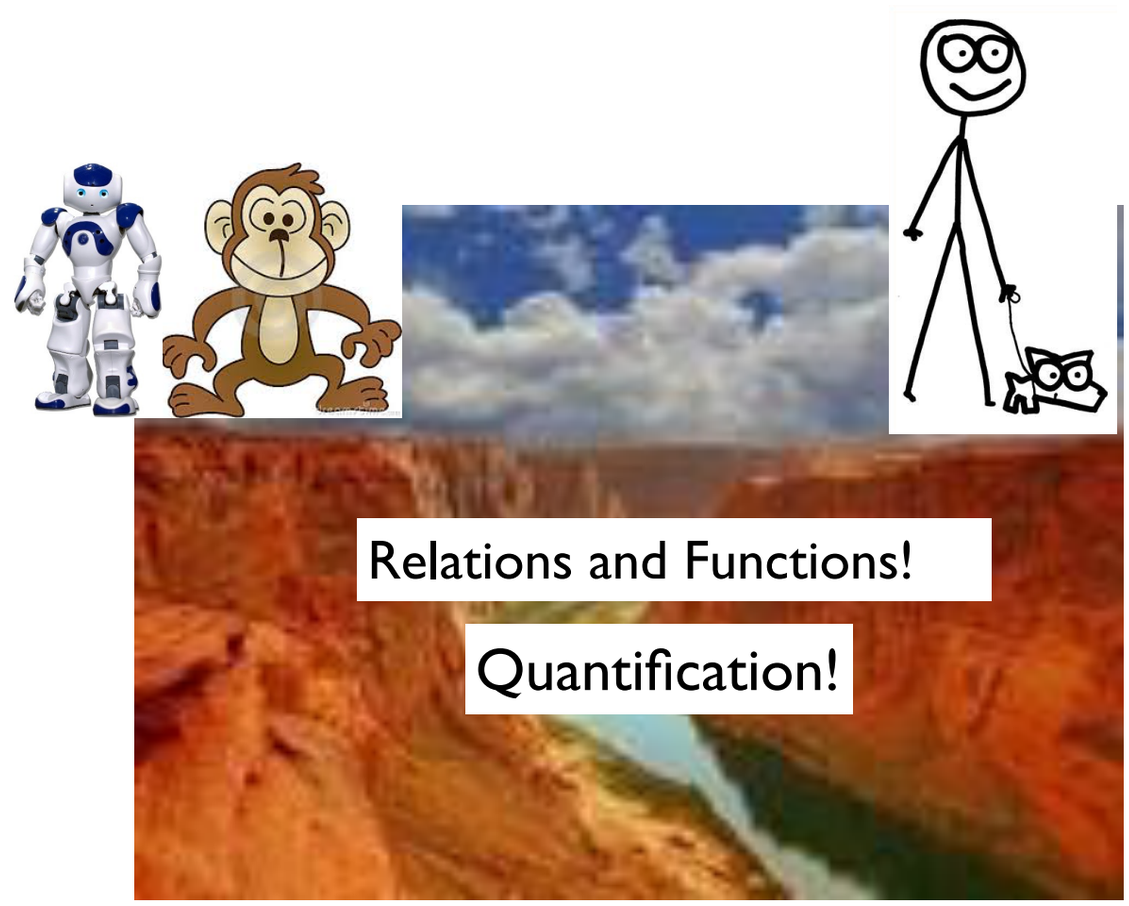
The Canyon of Discontinuity (or Darwin's Dread)



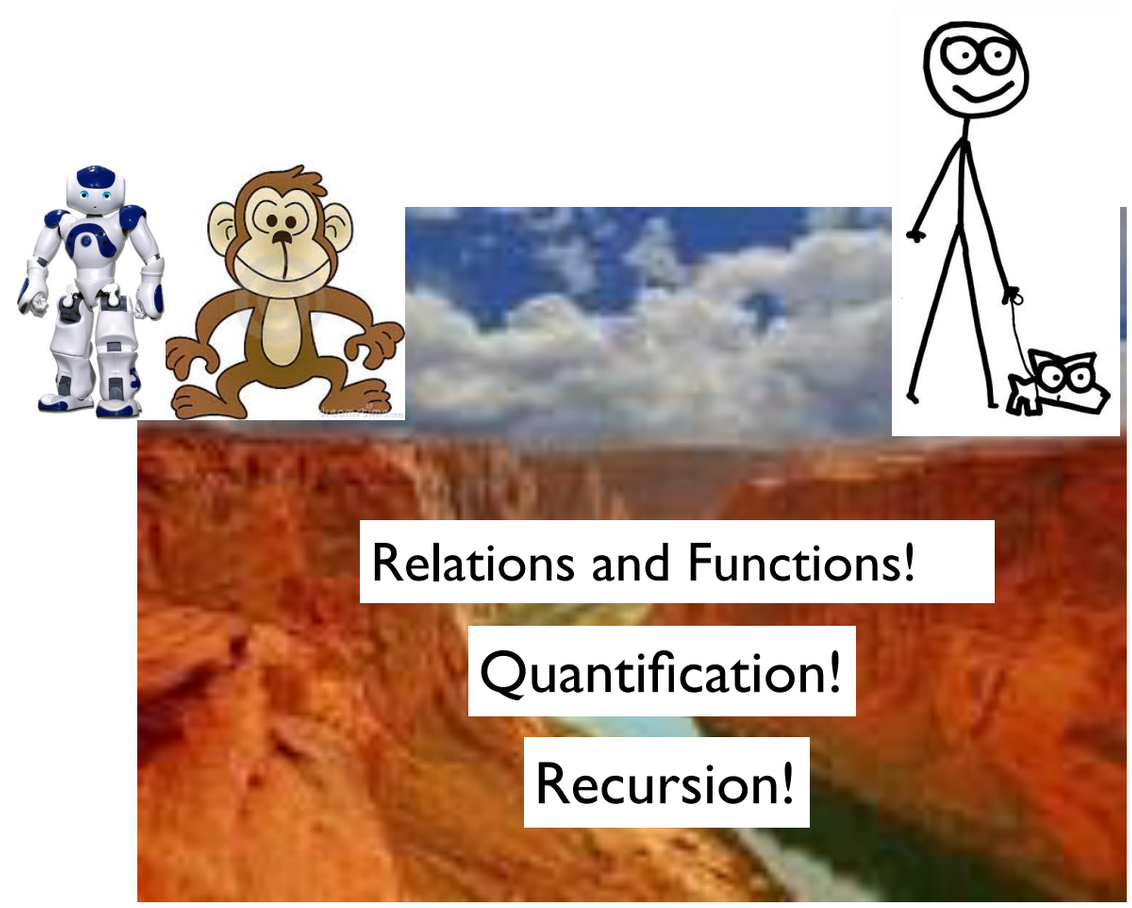
The Canyon of Discontinuity (or Darwin's Dread)



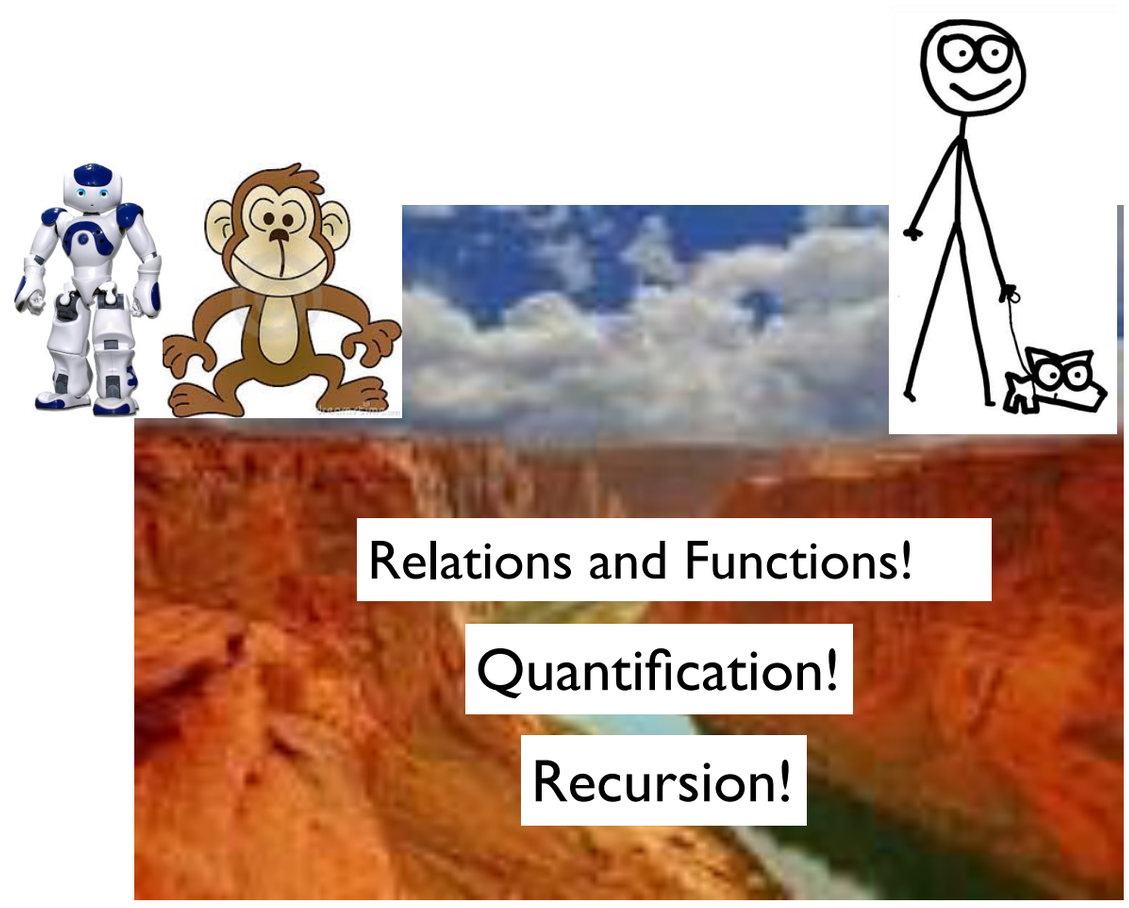
The Canyon of Discontinuity (or Darwin's Dread)



The Canyon of Discontinuity (or Darwin's Dread)

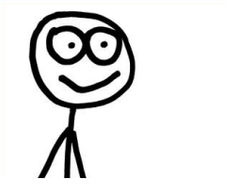


The Canyon of Discontinuity (or Darwin's Dread)

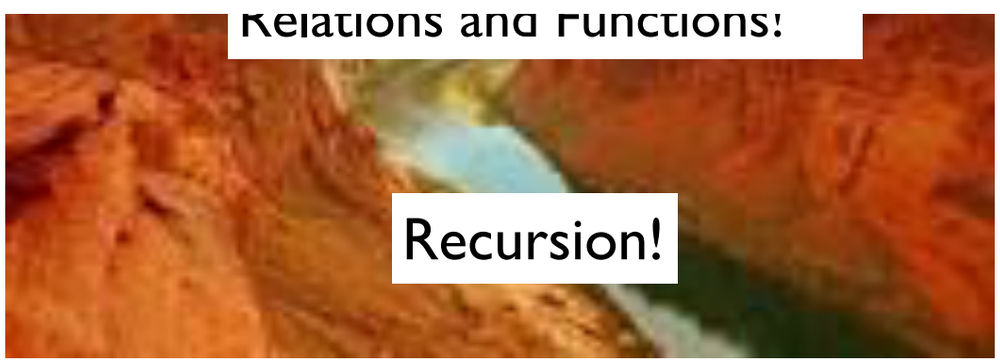


(Interesting paper: http://idiom.ucsd.edu/~ivano/SemBabble_old/LogicSeminar_15W/Material/Partee_2013_History-of-Quantifiers.pdf.)

The Canyon of Discontinuity (or Darwin's Dread)



Quantification!



(Interesting paper: http://idiom.ucsd.edu/~ivano/SemBabble_old/LogicSeminar_15W/Material/Partee_2013_History-of-Quantifiers.pdf.)

Karkooking Problem …

Everyone karkooks anyone who karkooks someone.

Alvin karkooks Bill.

Can you infer that everyone karkooks Bill?

ANSWER:

JUSTIFICATION:

Karkooking Problem …

Everyone karkooks anyone who karkooks someone.

Alvin karkooks Bill.

Can you infer that everyone karkooks Bill?

ANSWER:

JUSTIFICATION:

Relations and Functions!

Quantification!

Recursion!

Animal-Level AI

Super-Serious Human Cognitive Power

Serious Human Cognitive Power

Entscheidungsproblem

Mere Calculative Cognitive Power

Animal-Level AI

Analytical Hierarchy

Serious Human Cognitive Power

Entscheidungsproblem

Mere Calculative Cognitive Power

Animal-Level AI

Analytical Hierarchy

Arithmetical Hierarchy

Entscheidungsproblem

Mere Calculative Cognitive Power

Animal-Level AI

Analytical Hierarchy

Arithmetical Hierarchy

Entscheidungsproblem

Polynomial Hierarchy

Animal-Level AI

Analytical Hierarchy

Arithmetical Hierarchy

Entscheidungsproblem

Polynomial Hierarchy

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME} \subseteq \mathbf{EXPSPACE}$$

Animal-Level AI

Analytical Hierarchy

Arithmetical Hierarchy

$\vdots$

Π_2

Σ_2

Π_1

Σ_1

Σ_0

Entscheidungsproblem

Polynomial Hierarchy

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME} \subseteq \mathbf{EXPSPACE}$$

Animal-Level AI

Analytical Hierarchy

Arithmetical Hierarchy

$\vdots$

Π_2

Σ_2

Go: AlphaGo

Π_1

Σ_1

Σ_0

Entscheidungsproblem

Polynomial Hierarchy

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME} \subseteq \mathbf{EXPSPACE}$$

Animal-Level AI

Analytical Hierarchy

Arithmetical Hierarchy

$\vdots$

Π_2

Σ_2

Jeopardy!: **Watson**

Go: AlphaGo

Π_1

Σ_1

Σ_0

Entscheidungsproblem

Polynomial Hierarchy

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME} \subseteq \mathbf{EXPSPACE}$$

Animal-Level AI

Analytical Hierarchy

Arithmetical Hierarchy

$\vdots$

Π_2

Σ_2

Π_1

Σ_1

Σ_0

Chess: Deep Blue

Jeopardy!: **Watson**

Go: AlphaGo

Entscheidungsproblem

Polynomial Hierarchy

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME} \subseteq \mathbf{EXPSPACE}$$

Animal-Level AI

Analytical Hierarchy

Arithmetical Hierarchy

$\vdots$

Π_2

Σ_2

Π_1

Σ_1

Σ_0

Checkers: Chinook

Chess: Deep Blue

Jeopardy!: **Watson**

Go: AlphaGo

Entscheidungsproblem

Polynomial Hierarchy

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME} \subseteq \mathbf{EXPSPACE}$$

Animal-Level AI

Analytical Hierarchy

Arithmetical Hierarchy

$\vdots$

Π_2

Σ_2

Π_1

Σ_1

Σ_0

Entscheidungsproblem

Polynomial Hierarchy

Chess: Deep Blue

Checkers: Chinook

Go: AlphaGo

Jeopardy!: **Watson**

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME} \subseteq \mathbf{EXPSPACE}$$

Animal-Level AI

Analytical Hierarchy

Arithmetical Hierarchy

$\vdots$

Π_2

Σ_2

Π_1

Σ_1

Σ_0

Entscheidungsproblem

Polynomial Hierarchy

Chess: Deep Blue

Checkers: Chinook

Jeopardy!: **Watson**

Go: AlphaGo

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME} \subseteq \mathbf{EXPSPACE}$$

Animal-Level AI

Analytical Hierarchy

Arithmetical Hierarchy

Church

$\vdots$

Π_2

Σ_2

Π_1

Σ_1

Σ_0

Entscheidungsproblem

Polynomial Hierarchy

Chess: Deep Blue

Checkers: Chinook

Jeopardy!: **Watson**

Go: AlphaGo

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME} \subseteq \mathbf{EXPSPACE}$$

Animal-Level AI

Analytical Hierarchy

Arithmetical Hierarchy

Church

Turing

$\vdots$

Π_2

Σ_2

Π_1

Σ_1

Σ_0

Entscheidungsproblem

Polynomial Hierarchy

Chess: Deep Blue

Checkers: Chinook

Jeopardy!: **Watson**

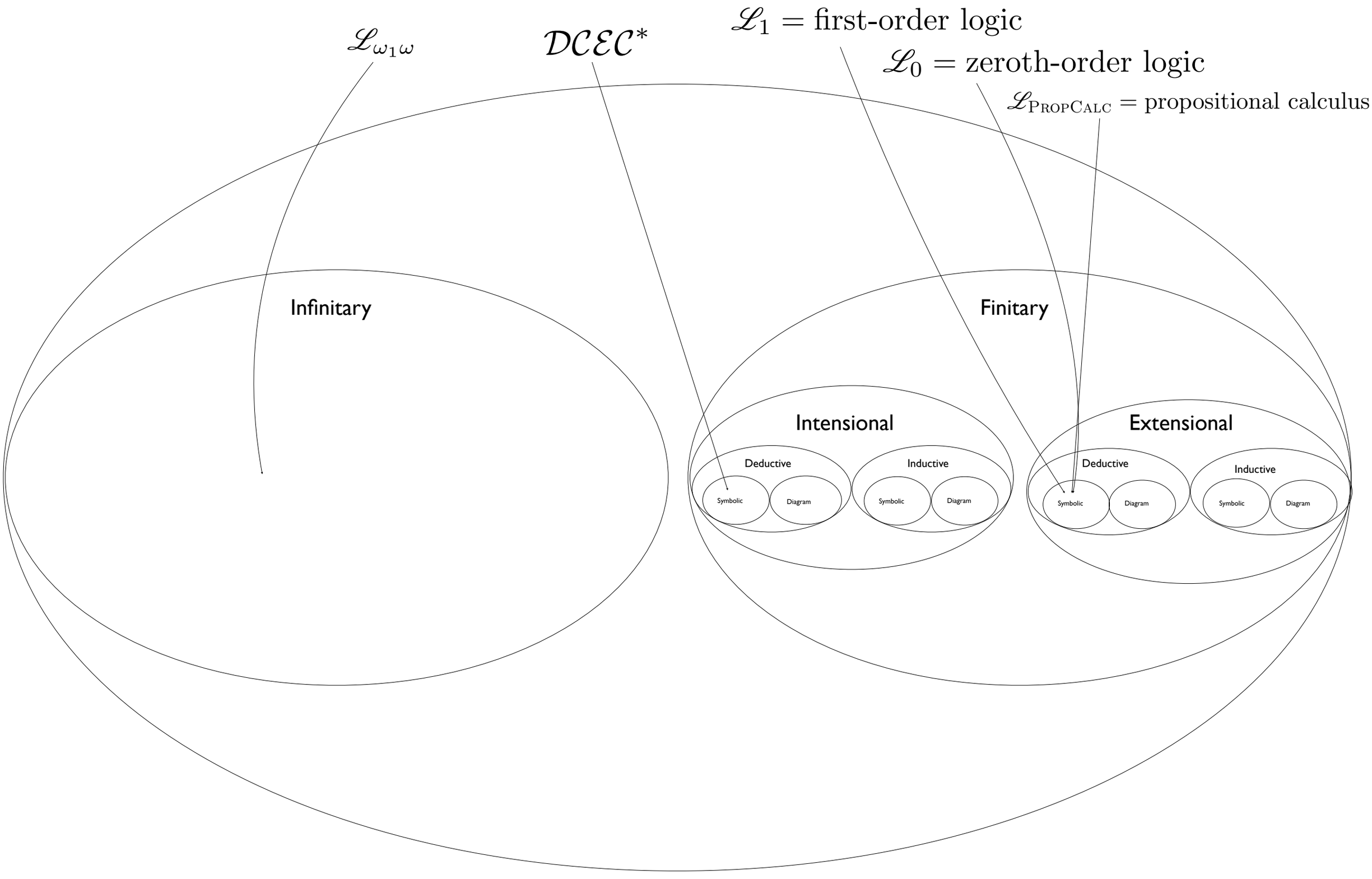
Go: AlphaGo

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{NEXPTIME} \subseteq \mathbf{EXPSPACE}$$

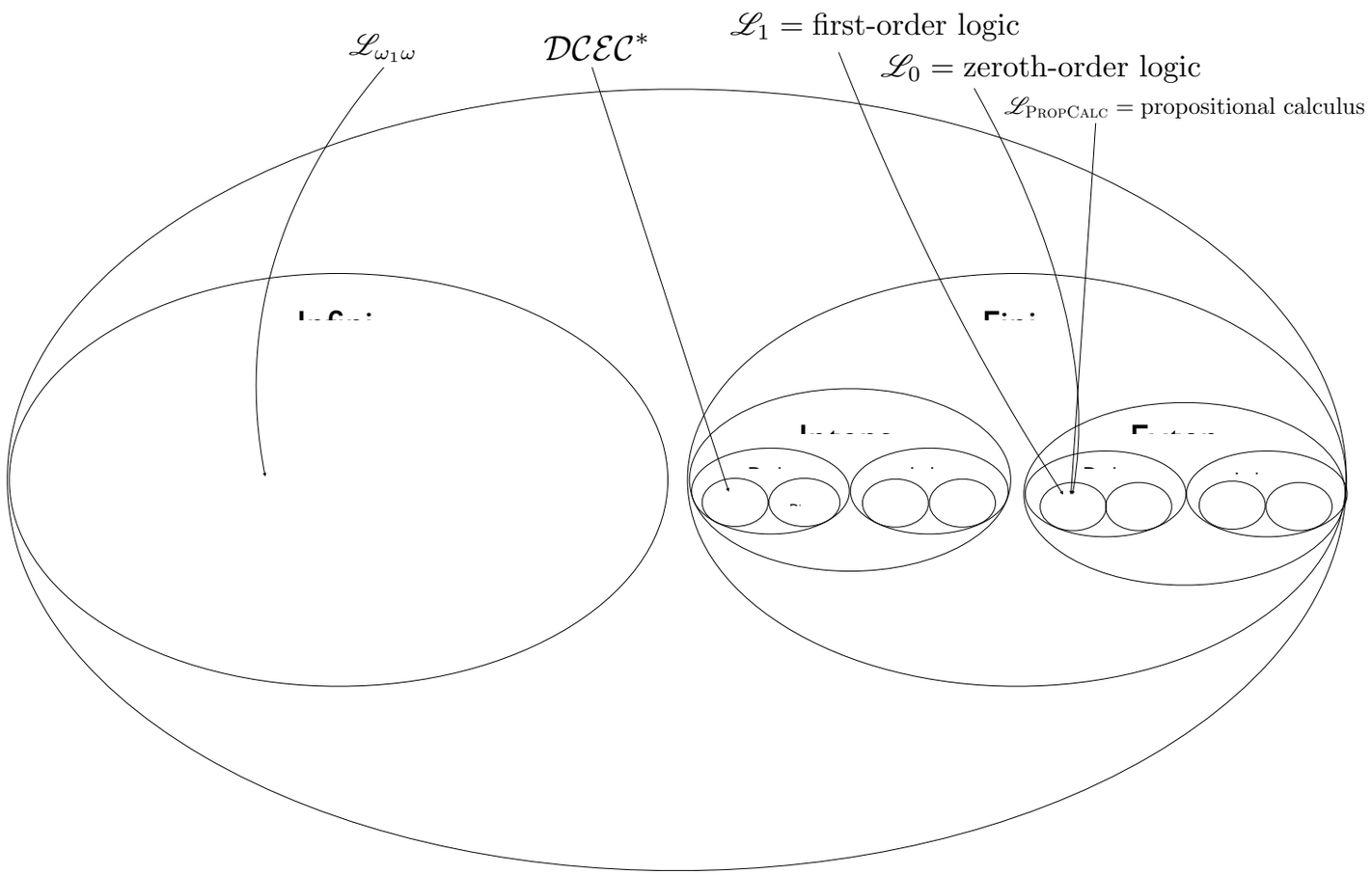
But what are the three things, in the triad, exactly?

Logic …

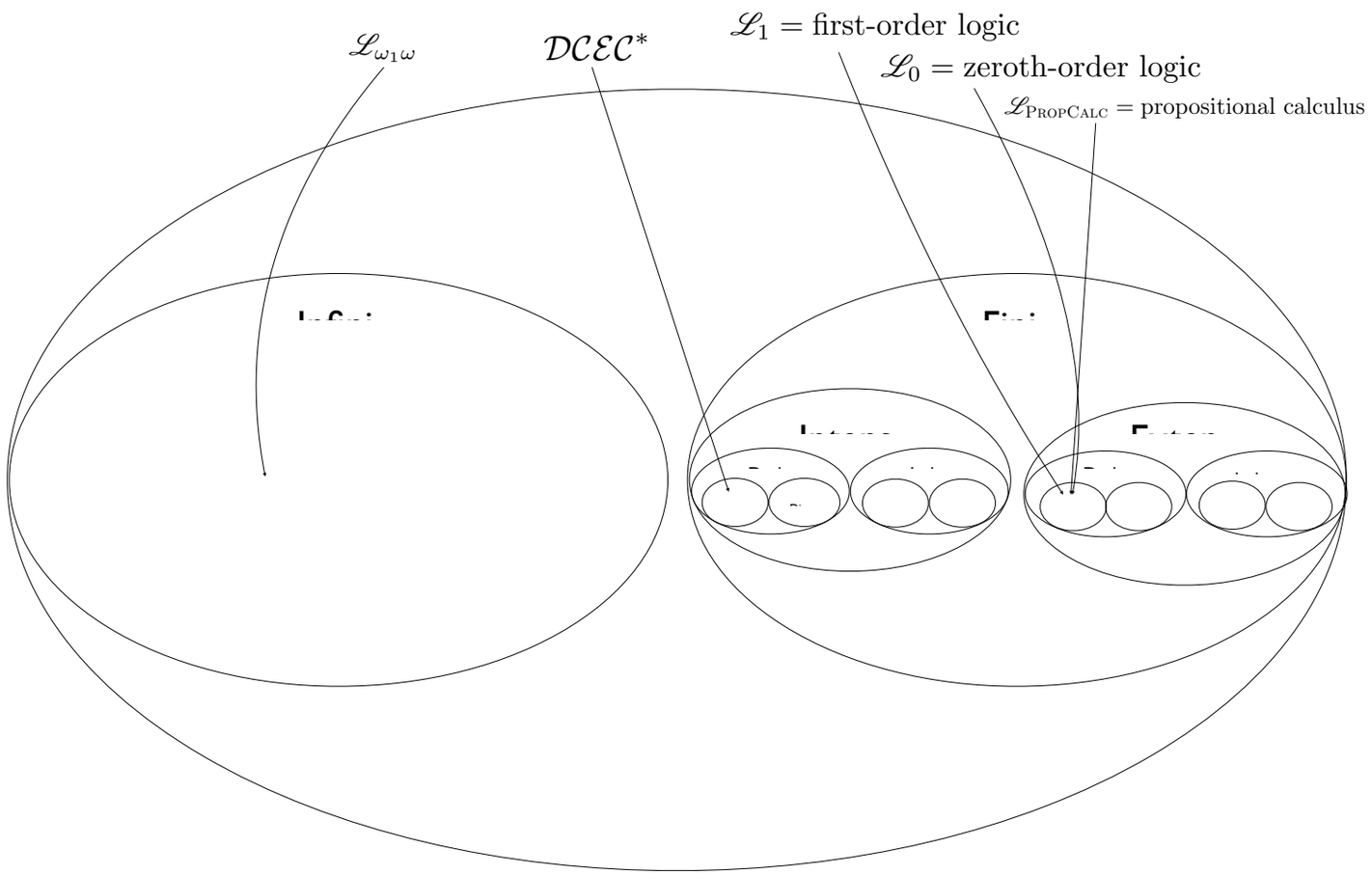
The Universe of Logics



The Universe of Logics

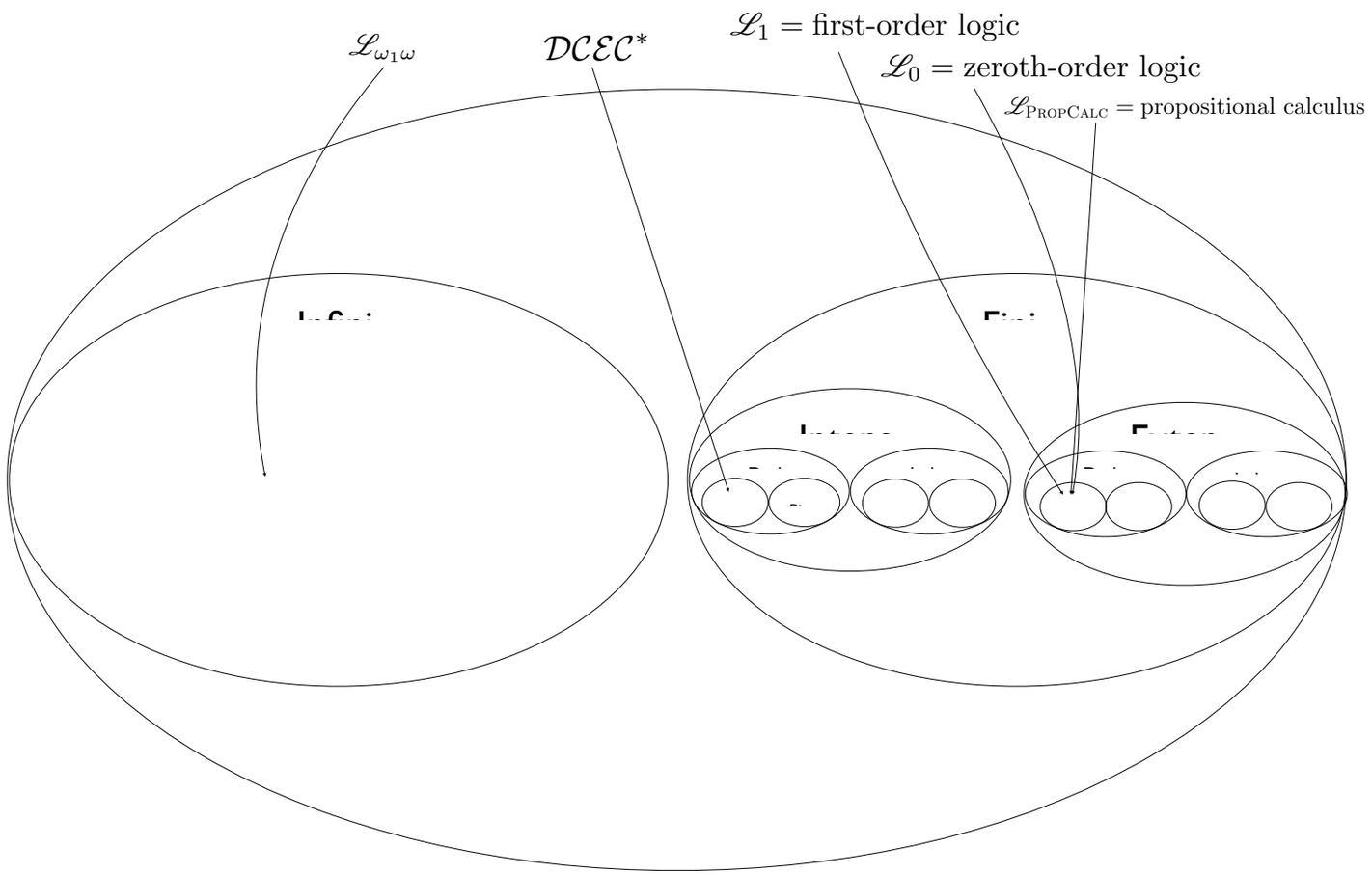


The Universe of Logics



$$\mathscr{L} := \langle L, I, S \rangle$$

The Universe of Logics



$$\mathscr{L} := \langle L, I, S \rangle$$

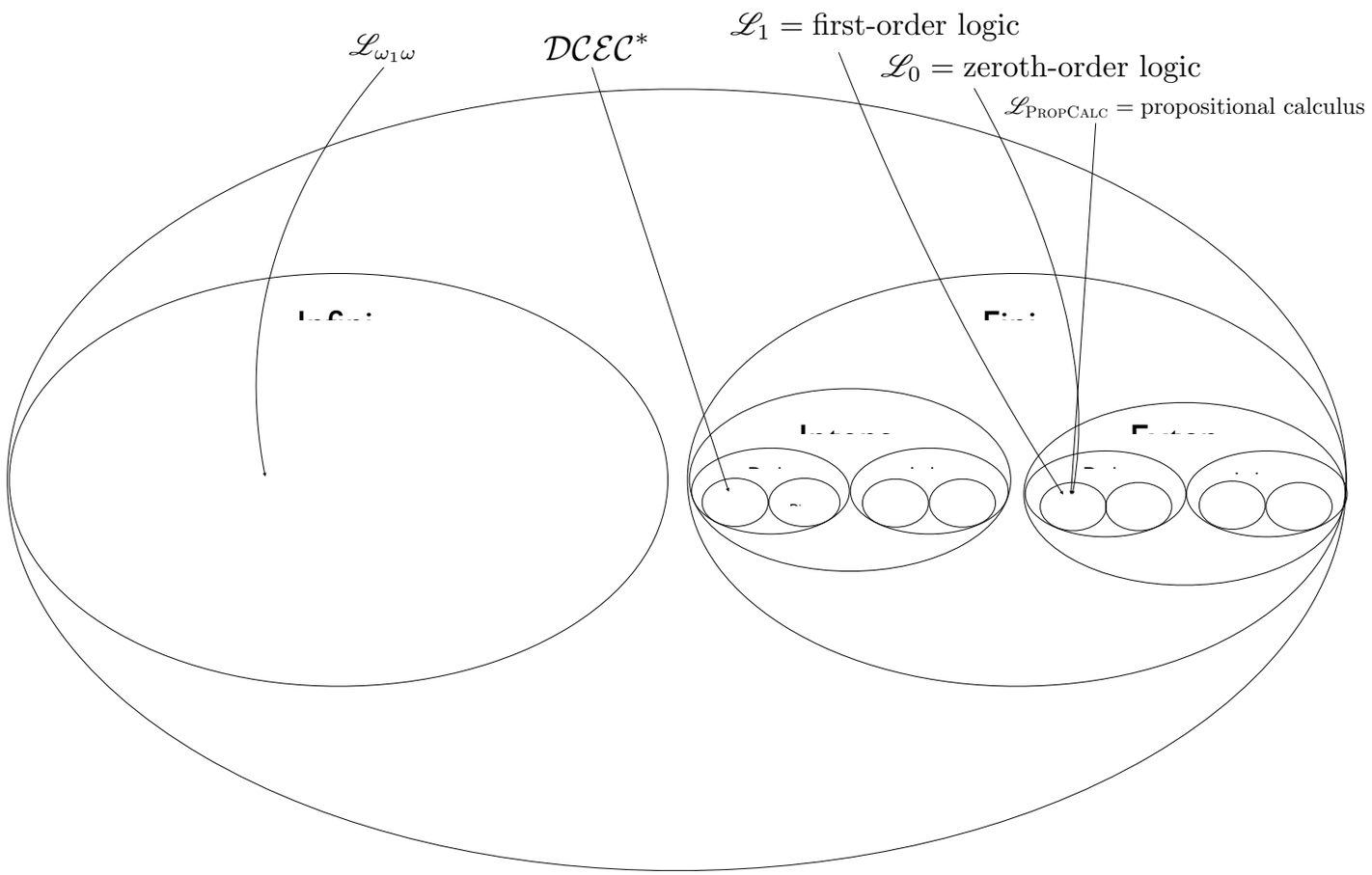
the logic

formal language

inference schemata

semantics

The Universe of Logics



$$\mathscr{L} := \langle L, I, \quad \rangle$$

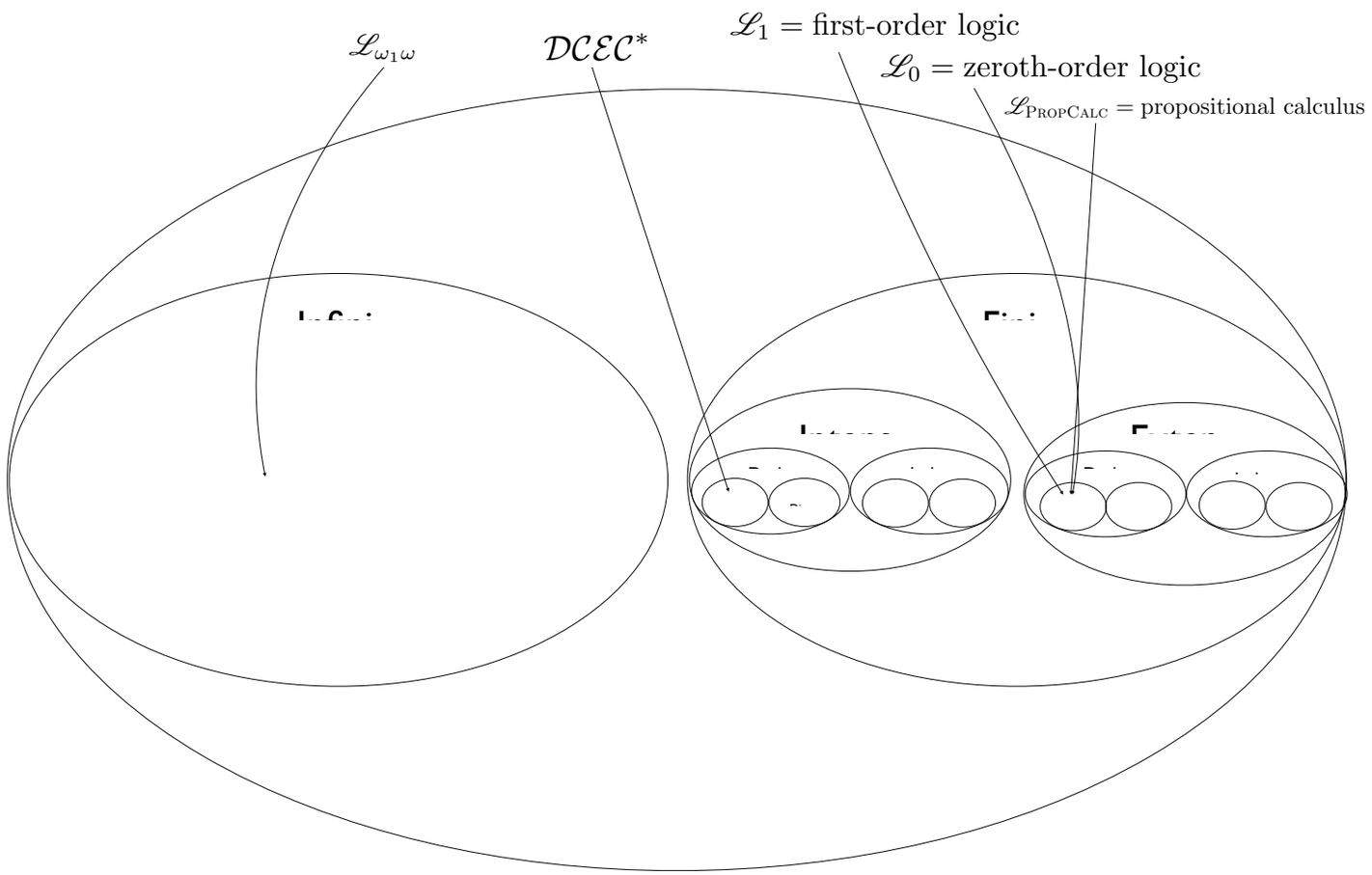
the logic

formal language

inference schemata

semantics

The Universe of Logics



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

the logic

formal language

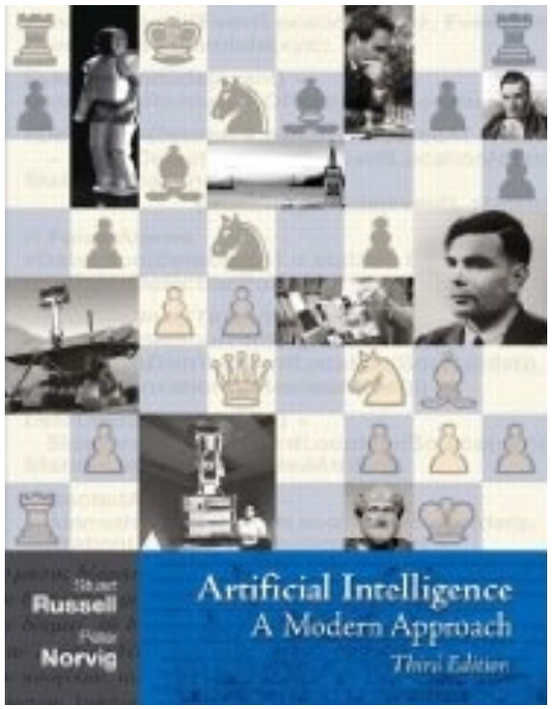
inference schemata

semantics

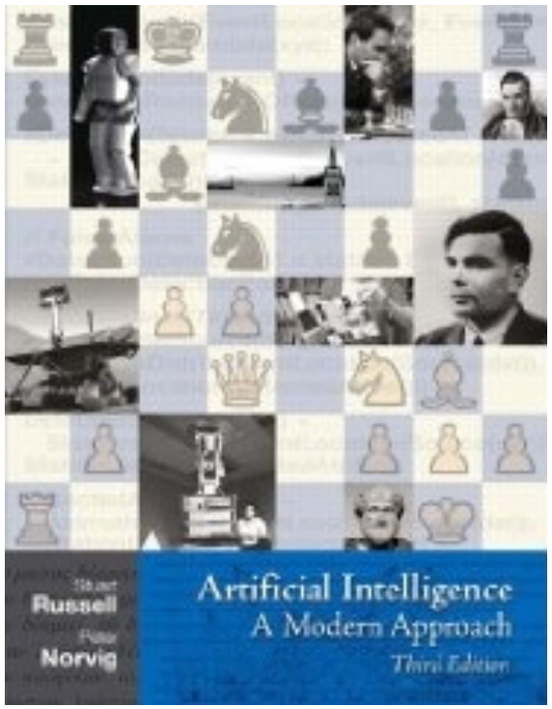
AI …

AI:

AI:


Stuart
Russell
Peter
Norvig
Artificial Intelligence
A Modern Approach
Third Edition

AI:



Stanford Encyclopedia of Philosophy

Browse About Support SEP

Search SEP

Entry Contents

Bibliography

Academic Tools

Friends PDF Preview

Author and Citation Info

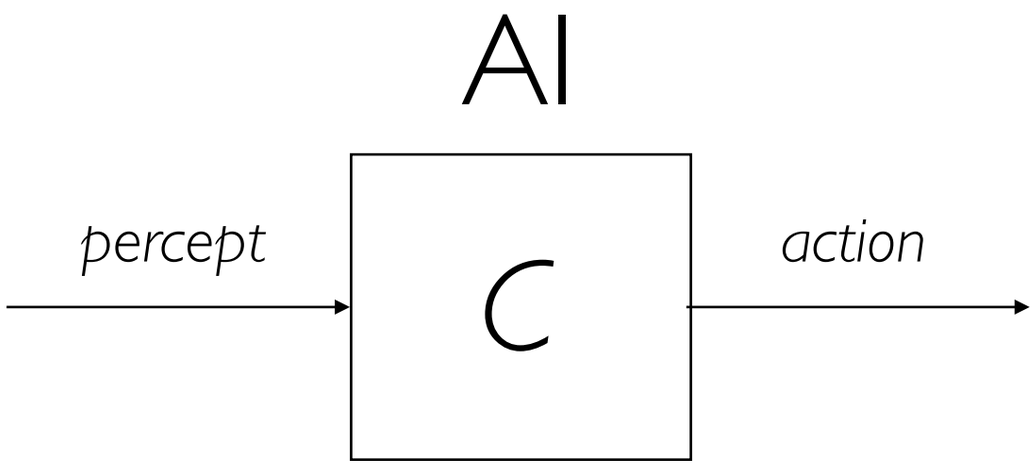
Back to Top

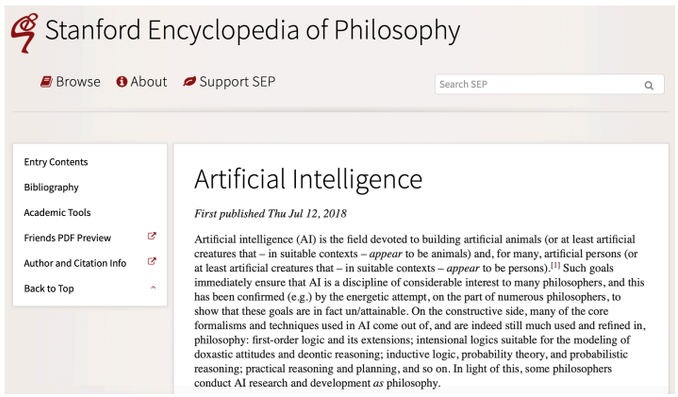
Artificial Intelligence

First published Thu Jul 12, 2018

Artificial intelligence (AI) is the field devoted to building artificial animals (or at least artificial creatures that – in suitable contexts – *appear* to be animals) and, for many, artificial persons (or at least artificial creatures that – in suitable contexts – *appear* to be persons).[1] Such goals immediately ensure that AI is a discipline of considerable interest to many philosophers, and this has been confirmed (e.g.) by the energetic attempt, on the part of numerous philosophers, to show that these goals are in fact un/attainable. On the constructive side, many of the core formalisms and techniques used in AI come out of, and are indeed still much used and refined in, philosophy: first-order logic and its extensions; intensional logics suitable for the modeling of doxastic attitudes and deontic reasoning; inductive logic, probability theory, and probabilistic reasoning; practical reasoning and planning, and so on. In light of this, some philosophers conduct AI research and development *as* philosophy.

AI:




Stanford Encyclopedia of Philosophy

Browse About Support SEP

Search SEP

Entry Contents

Bibliography

Academic Tools

Friends PDF Preview

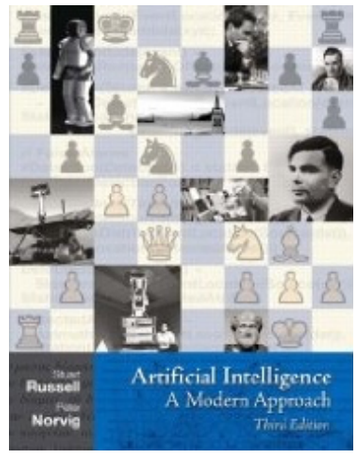
Author and Citation Info

Back to Top

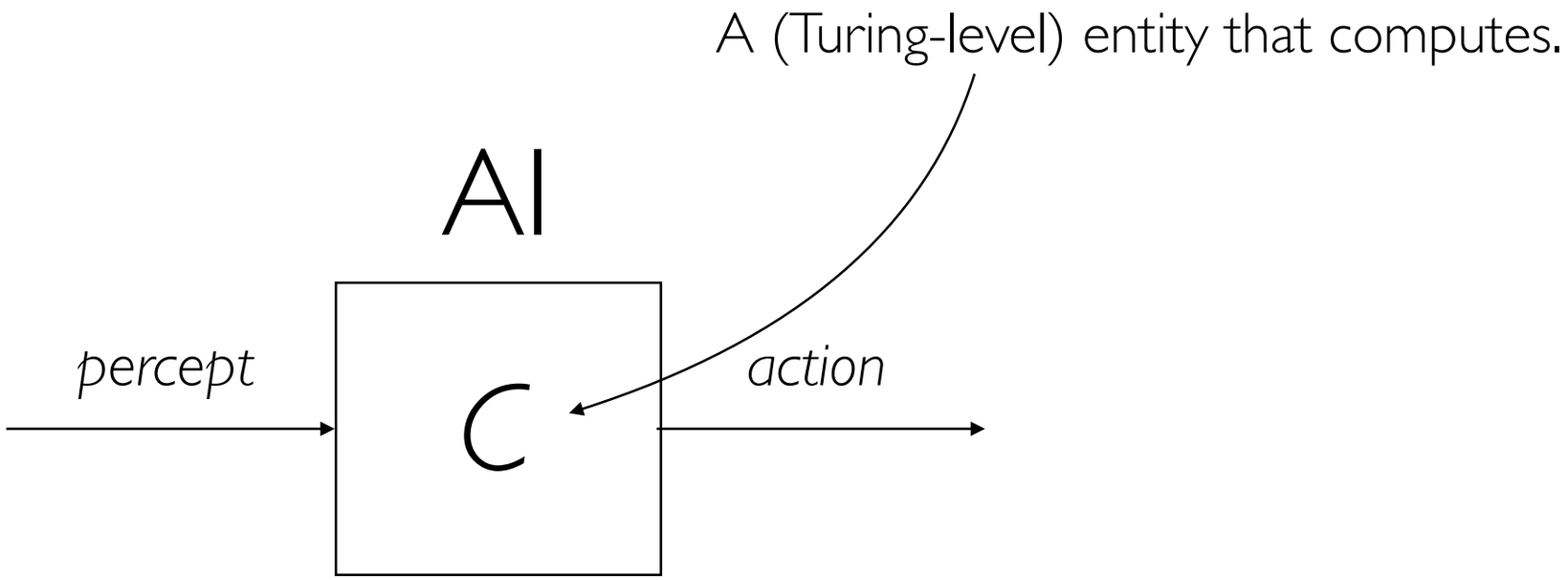
Artificial Intelligence

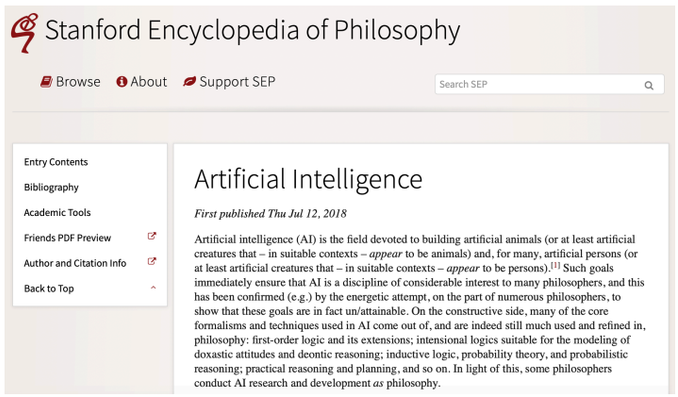
First published Thu Jul 12, 2018

Artificial intelligence (AI) is the field devoted to building artificial animals (or at least artificial creatures that – in suitable contexts – *appear* to be animals) and, for many, artificial persons (or at least artificial creatures that – in suitable contexts – *appear* to be persons).[1] Such goals immediately ensure that AI is a discipline of considerable interest to many philosophers, and this has been confirmed (e.g.) by the energetic attempt, on the part of numerous philosophers, to show that these goals are in fact un/attainable. On the constructive side, many of the core formalisms and techniques used in AI come out of, and are indeed still much used and refined in, philosophy: first-order logic and its extensions; intensional logics suitable for the modeling of doxastic attitudes and deontic reasoning; inductive logic, probability theory, and probabilistic reasoning; practical reasoning and planning, and so on. In light of this, some philosophers conduct AI research and development *as* philosophy.



AI:




Stanford Encyclopedia of Philosophy

Browse About Support SEP

Search SEP

Entry Contents

Bibliography

Academic Tools

Friends PDF Preview

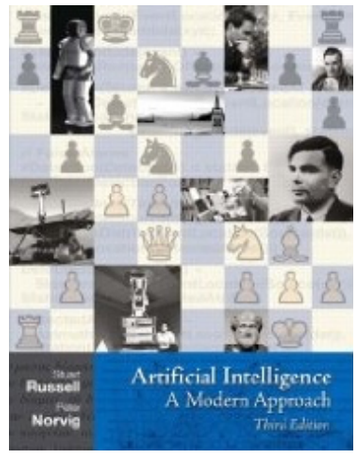
Author and Citation Info

Back to Top

Artificial Intelligence

First published Thu Jul 12, 2018

Artificial intelligence (AI) is the field devoted to building artificial animals (or at least artificial creatures that – in suitable contexts – *appear* to be animals) and, for many, artificial persons (or at least artificial creatures that – in suitable contexts – *appear* to be persons).[1] Such goals immediately ensure that AI is a discipline of considerable interest to many philosophers, and this has been confirmed (e.g.) by the energetic attempt, on the part of numerous philosophers, to show that these goals are in fact un/attainable. On the constructive side, many of the core formalisms and techniques used in AI come out of, and are indeed still much used and refined in, philosophy: first-order logic and its extensions; intensional logics suitable for the modeling of doxastic attitudes and deontic reasoning; inductive logic, probability theory, and probabilistic reasoning; practical reasoning and planning, and so on. In light of this, some philosophers conduct AI research and development *as* philosophy.

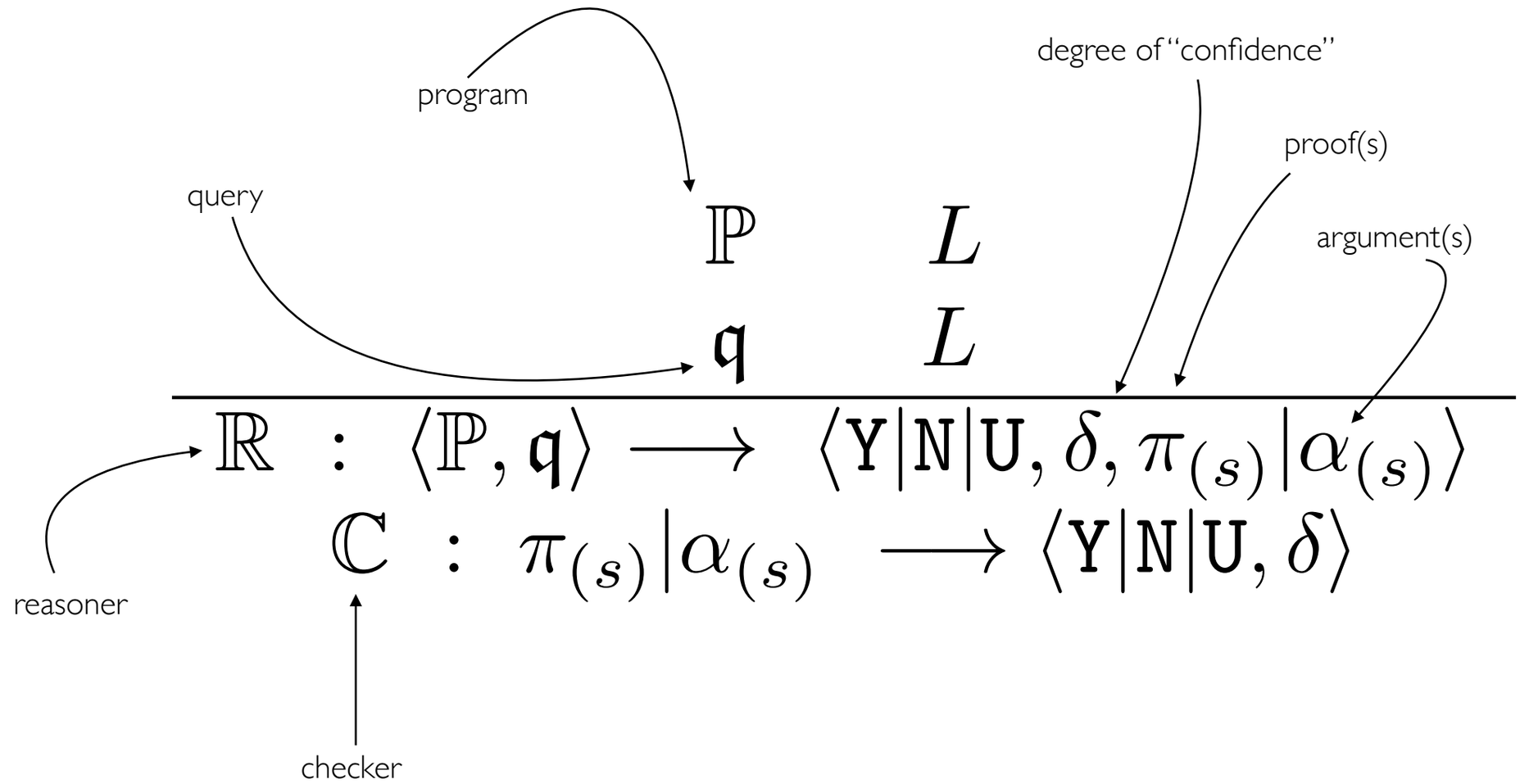


(Pure General) Logic Programming …

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

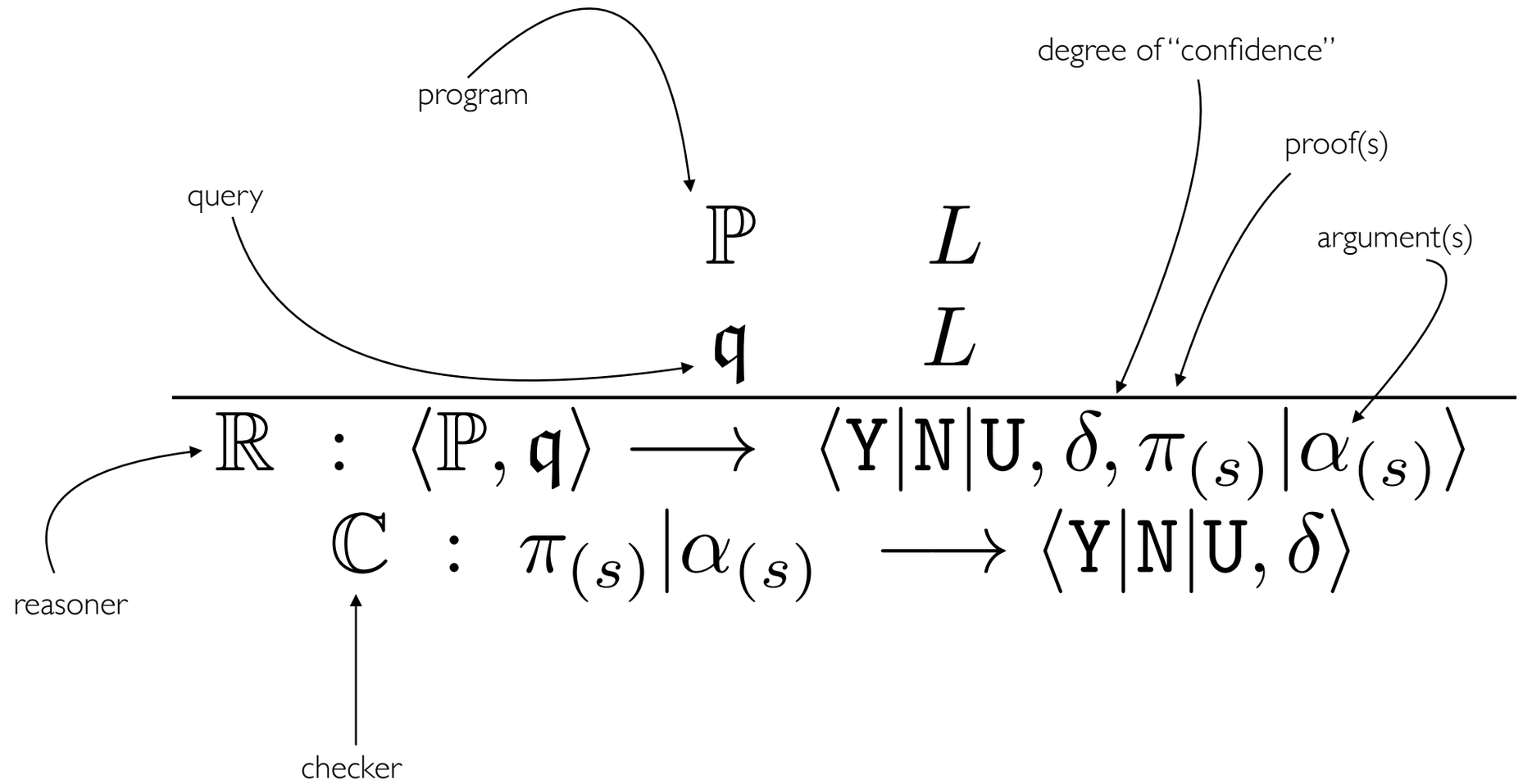
$$\frac{\begin{array}{cc} \mathbb{P} & L \\ \mathfrak{q} & L \end{array}}{\begin{array}{c} \mathbb{R} \ : \ \langle \mathbb{P}, \mathfrak{q} \rangle \longrightarrow \langle \mathtt{Y} | \mathtt{N} | \mathtt{U}, \delta, \pi_{(s)} | \alpha_{(s)} \rangle \\ \mathbb{C} \ : \ \pi_{(s)} | \alpha_{(s)} \longrightarrow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

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



For just "logic programming," and a vintage approach that goes back to circa 1970, restrict this to a FOL or a fragment thereof, and use resolution as the only inference schema.

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



For just "logic programming," and a vintage approach that goes back to circa 1970, restrict this to a FOL or a fragment thereof, and use resolution as the only inference schema.

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

program

query

degree of "confidence"

proof(s)

argument(s)

$$\frac{\mathbb{P} \quad L \qquad \mathfrak{q} \quad L}{\mathbb{R} \ : \ \langle \mathbb{P}, \mathfrak{q} \rangle \longrightarrow \langle \mathtt{Y} | \mathtt{N} | \mathtt{U}, \delta, \pi_{(s)} | \alpha_{(s)} \rangle}$$

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

reasoner

checker

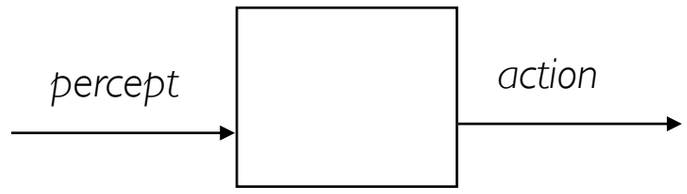
Resurrection of The Triad

The Triad Resurrected & Rebuilt, & Better

Logic

$\mathcal{L}$

AI



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

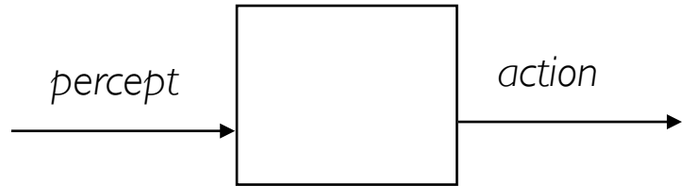
Pure General Logic Programming

The Triad Resurrected & Rebuilt, & Better

Logic

$\mathcal{L}$

AI



$$\boldsymbol{\mathcal{L}} \coloneqq \langle L, \mathcal{I} \rangle \qquad \frac{\begin{array}{cc} \mathbb{P} & L \\ \mathfrak{q} & L \end{array}}{\begin{array}{c} \mathbb{R} \ : \ \langle \mathbb{P}, \mathfrak{q} \rangle \longrightarrow \langle \mathtt{Y} | \mathtt{N} | \mathtt{U}, \delta, \pi_{(s)} | \alpha_{(s)} \rangle \\ \mathbb{C} \ : \ \pi_{(s)} | \alpha_{(s)} \longrightarrow \langle \mathtt{Y} | \mathtt{N} | \mathtt{U}, \delta \rangle \end{array}}$$

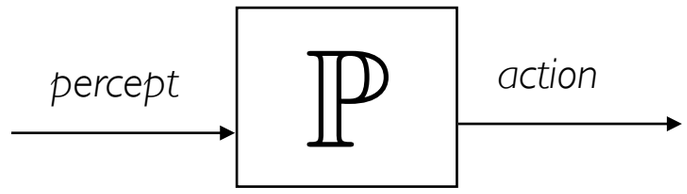
Pure General Logic Programming

The Triad Resurrected & Rebuilt, & Better

Logic

$\mathcal{L}$

AI



$$\boldsymbol{\mathcal{L}} := \langle L, \mathcal{I} \rangle \qquad \frac{\begin{array}{cc} \mathbb{P} & L \\ \mathfrak{q} & L \end{array}}{\begin{array}{c} \mathbb{R} \ : \ \langle \mathbb{P}, \mathfrak{q} \rangle \longrightarrow \langle \mathtt{Y} | \mathtt{N} | \mathtt{U}, \delta, \pi_{(s)} | \alpha_{(s)} \rangle \\ \mathbb{C} \ : \ \pi_{(s)} | \alpha_{(s)} \longrightarrow \langle \mathtt{Y} | \mathtt{N} | \mathtt{U}, \delta \rangle \end{array}}$$

Pure General Logic Programming