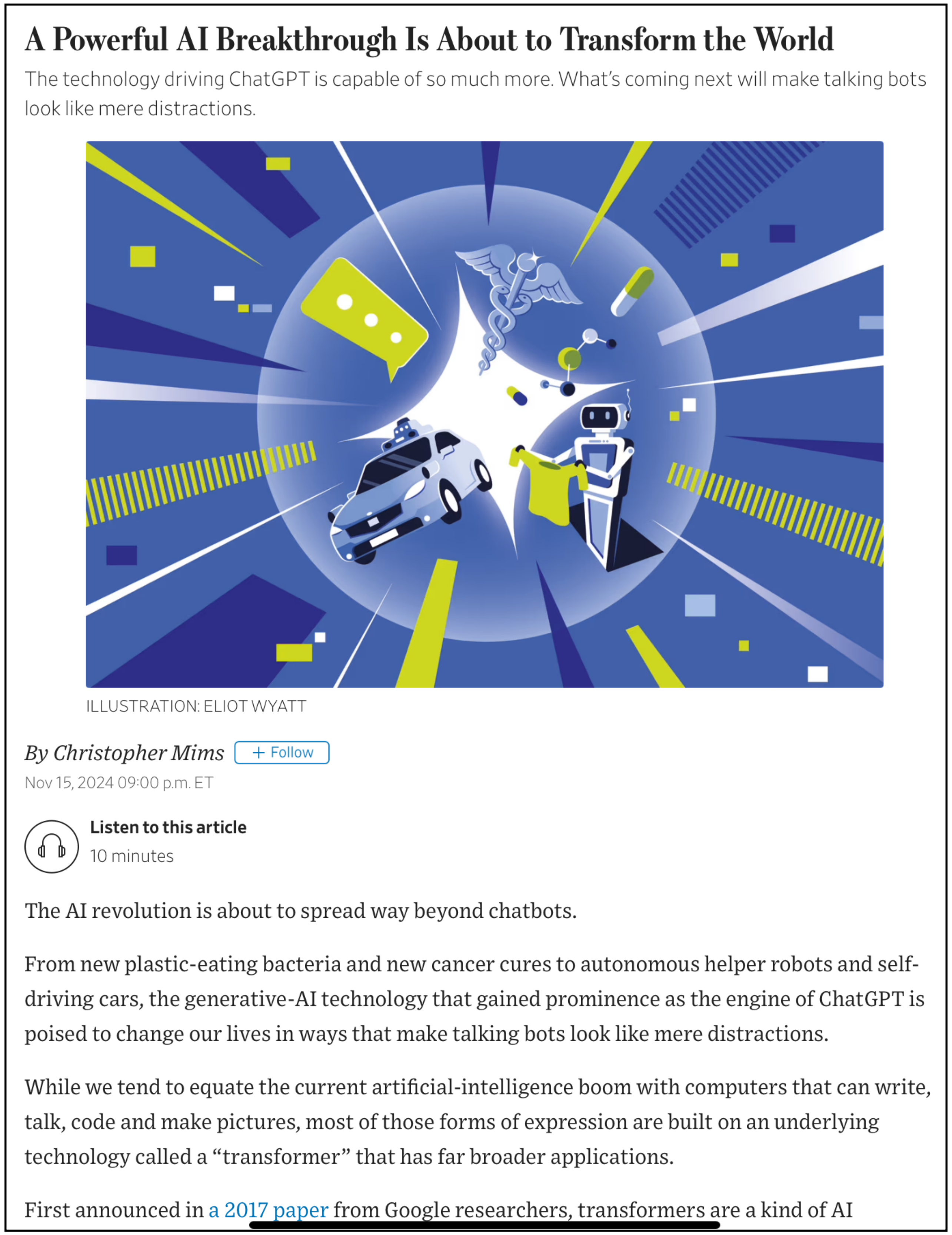
Are Programs Uniformly Finite? A Size-based Progression in the Context of Pure General Logic Programming of Programs Says: "No"

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

ILBAI
Nov 18 2025
`ver 1118241351NY`

Logic-&-AI In The News



A Powerful AI Breakthrough Is About to Transform the World

The technology driving ChatGPT is capable of so much more. What's coming next will make talking bots look like mere distractions.

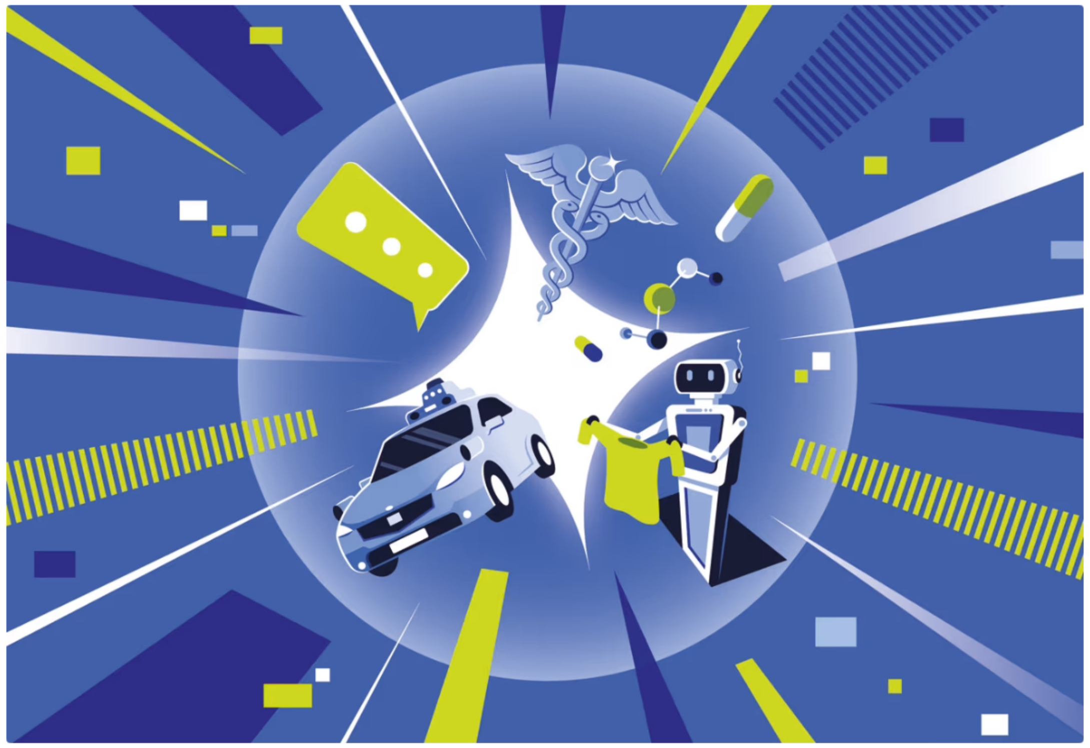


ILLUSTRATION: ELIOT WYATT

By Christopher Mims + Follow

Nov 15, 2024 09:00 p.m. ET

Listen to this article
10 minutes

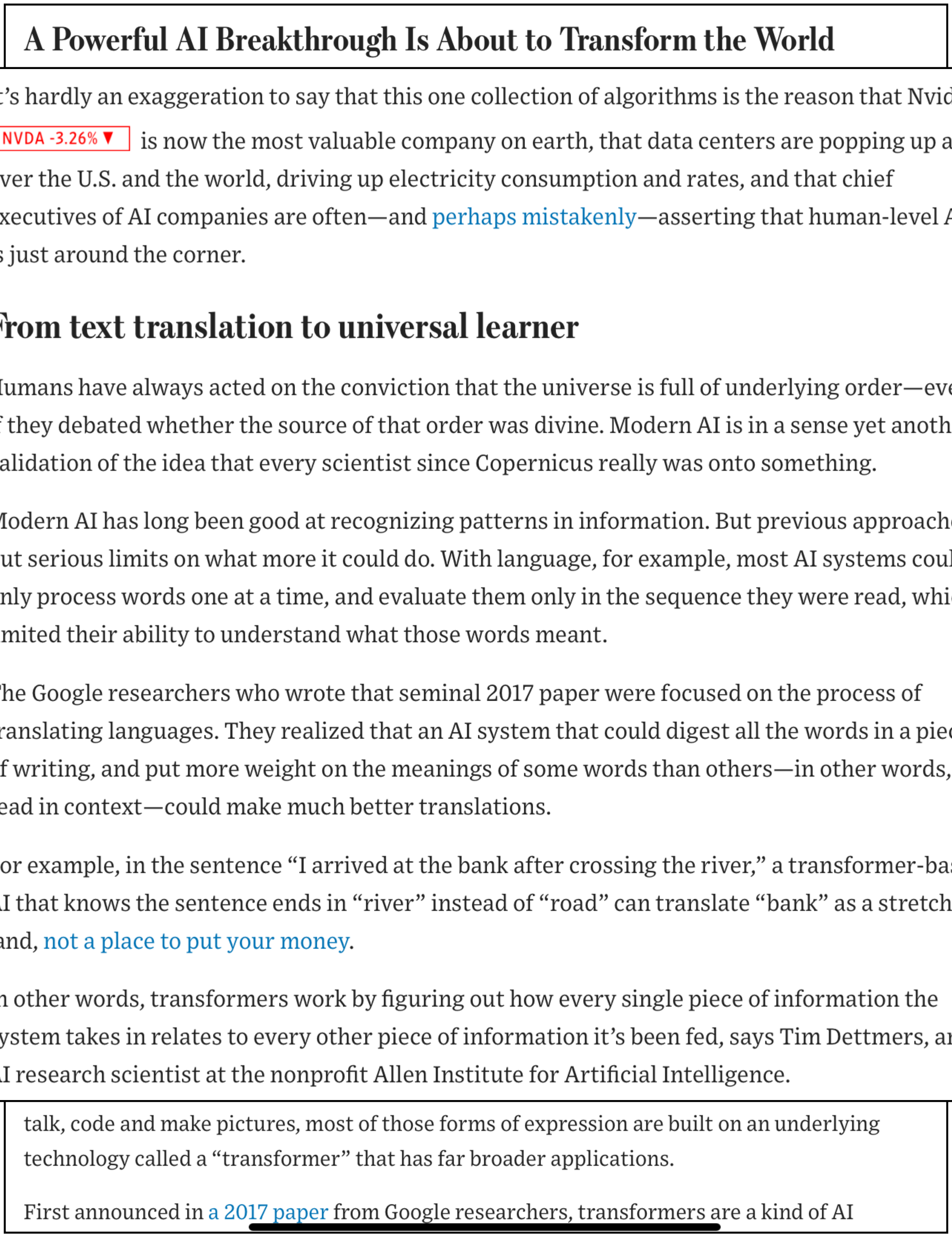
The AI revolution is about to spread way beyond chatbots.

From new plastic-eating bacteria and new cancer cures to autonomous helper robots and self-driving cars, the generative-AI technology that gained prominence as the engine of ChatGPT is poised to change our lives in ways that make talking bots look like mere distractions.

While we tend to equate the current artificial-intelligence boom with computers that can write, talk, code and make pictures, most of those forms of expression are built on an underlying technology called a "transformer" that has far broader applications.

First announced in a 2017 paper from Google researchers, transformers are a kind of AI

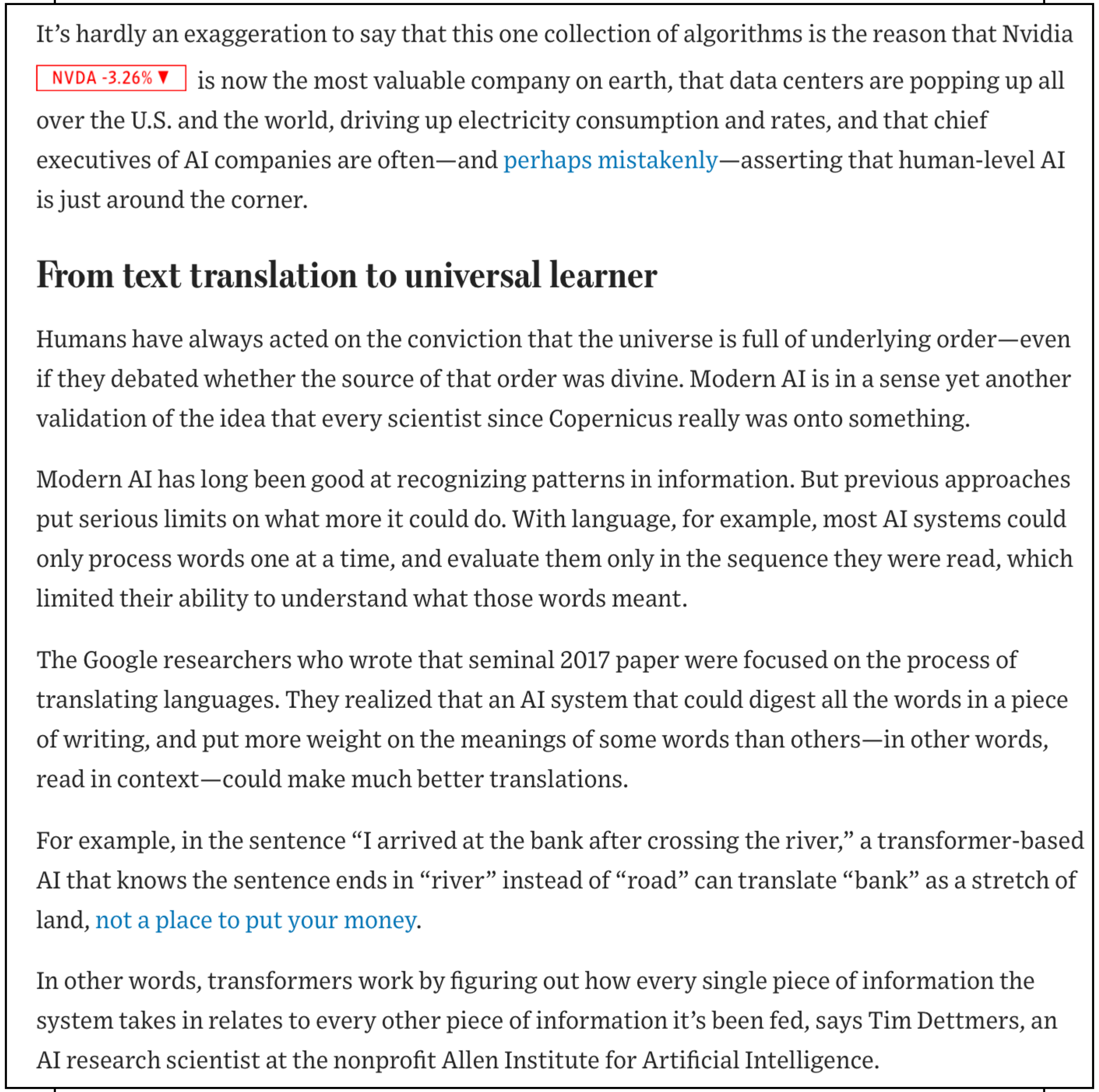
Logic-&-AI In The News



A Powerful AI Breakthrough Is About to Transform the World

talk, code and make pictures, most of those forms of expression are built on an underlying technology called a "transformer" that has far broader applications.

First announced in a 2017 paper from Google researchers, transformers are a kind of AI



It's hardly an exaggeration to say that this one collection of algorithms is the reason that Nvidia NVDA -3.26% ▼ is now the most valuable company on earth, that data centers are popping up all over the U.S. and the world, driving up electricity consumption and rates, and that chief executives of AI companies are often—and perhaps mistakenly—asserting that human-level AI is just around the corner.

From text translation to universal learner

Humans have always acted on the conviction that the universe is full of underlying order—even if they debated whether the source of that order was divine. Modern AI is in a sense yet another validation of the idea that every scientist since Copernicus really was onto something.

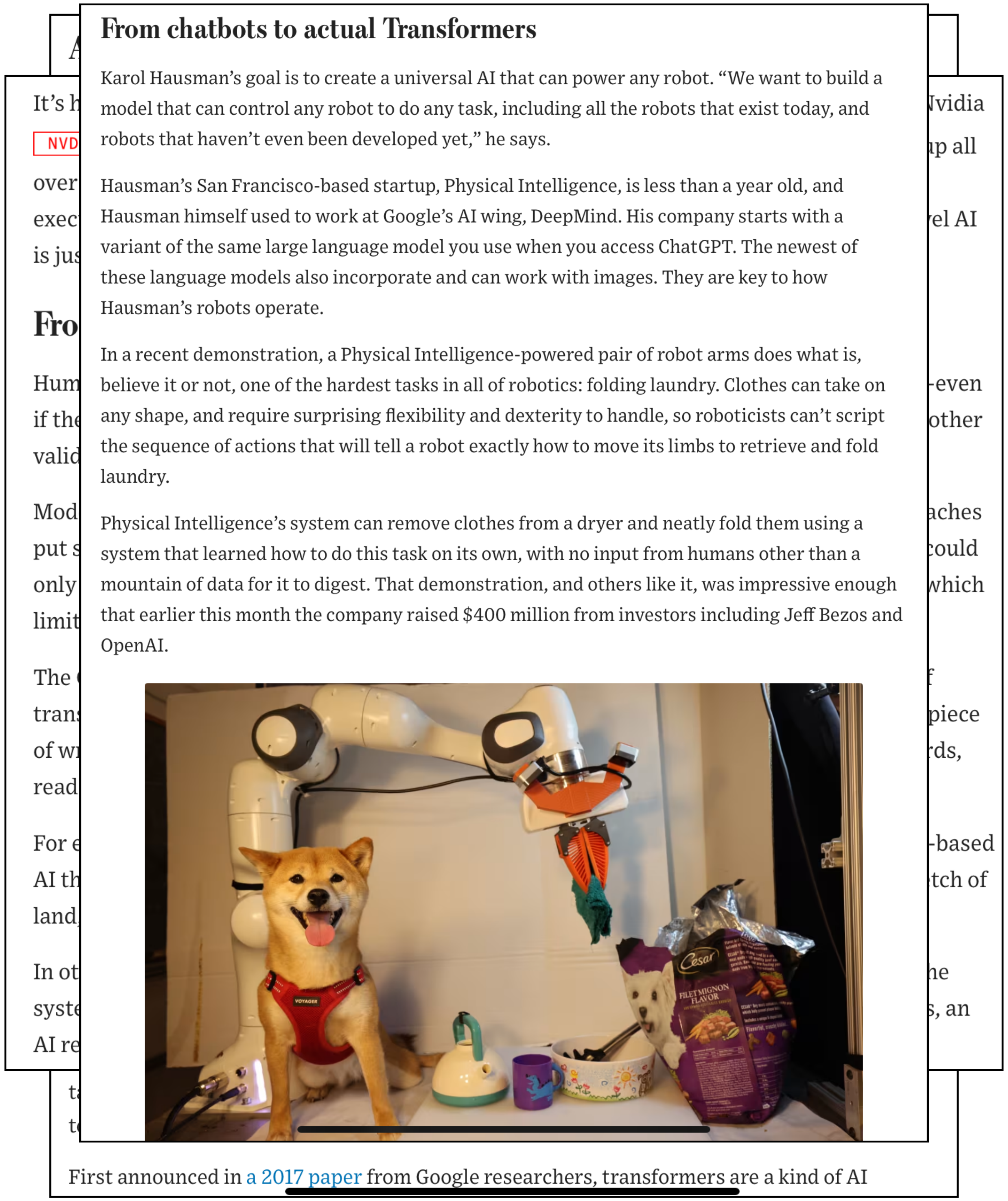
Modern AI has long been good at recognizing patterns in information. But previous approaches put serious limits on what more it could do. With language, for example, most AI systems could only process words one at a time, and evaluate them only in the sequence they were read, which limited their ability to understand what those words meant.

The Google researchers who wrote that seminal 2017 paper were focused on the process of translating languages. They realized that an AI system that could digest all the words in a piece of writing, and put more weight on the meanings of some words than others—in other words, read in context—could make much better translations.

For example, in the sentence "I arrived at the bank after crossing the river," a transformer-based AI that knows the sentence ends in "river" instead of "road" can translate "bank" as a stretch of land, not a place to put your money.

In other words, transformers work by figuring out how every single piece of information the system takes in relates to every other piece of information it's been fed, says Tim Dettmers, an AI research scientist at the nonprofit Allen Institute for Artificial Intelligence.

Logic-&-AI In The News



From chatbots to actual Transformers

Karol Hausman's goal is to create a universal AI that can power any robot. "We want to build a model that can control any robot to do any task, including all the robots that exist today, and robots that haven't even been developed yet," he says.

Hausman's San Francisco-based startup, Physical Intelligence, is less than a year old, and Hausman himself used to work at Google's AI wing, DeepMind. His company starts with a variant of the same large language model you use when you access ChatGPT. The newest of these language models also incorporate and can work with images. They are key to how Hausman's robots operate.

In a recent demonstration, a Physical Intelligence-powered pair of robot arms does what is, believe it or not, one of the hardest tasks in all of robotics: folding laundry. Clothes can take on any shape, and require surprising flexibility and dexterity to handle, so roboticists can't script the sequence of actions that will tell a robot exactly how to move its limbs to retrieve and fold laundry.

Physical Intelligence's system can remove clothes from a dryer and neatly fold them using a system that learned how to do this task on its own, with no input from humans other than a mountain of data for it to digest. That demonstration, and others like it, was impressive enough that earlier this month the company raised $400 million from investors including Jeff Bezos and OpenAI.

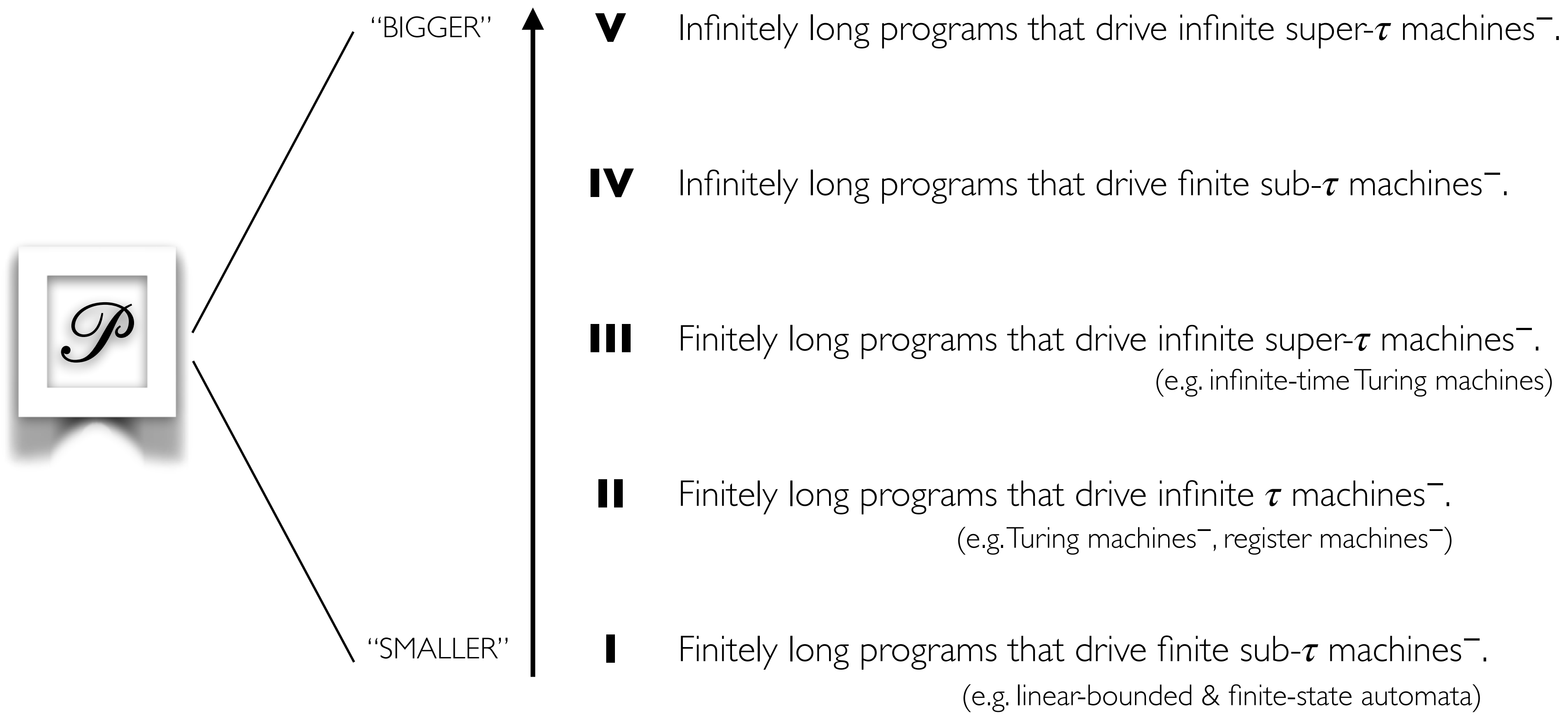
First announced in a 2017 paper from Google researchers, transformers are a kind of AI

Initial Specimen

$$\mathfrak{p}_1$$

$$s_1 \ b \ 1 \ s_2$$

$$s_2 \ 1 \ r \ s_1$$



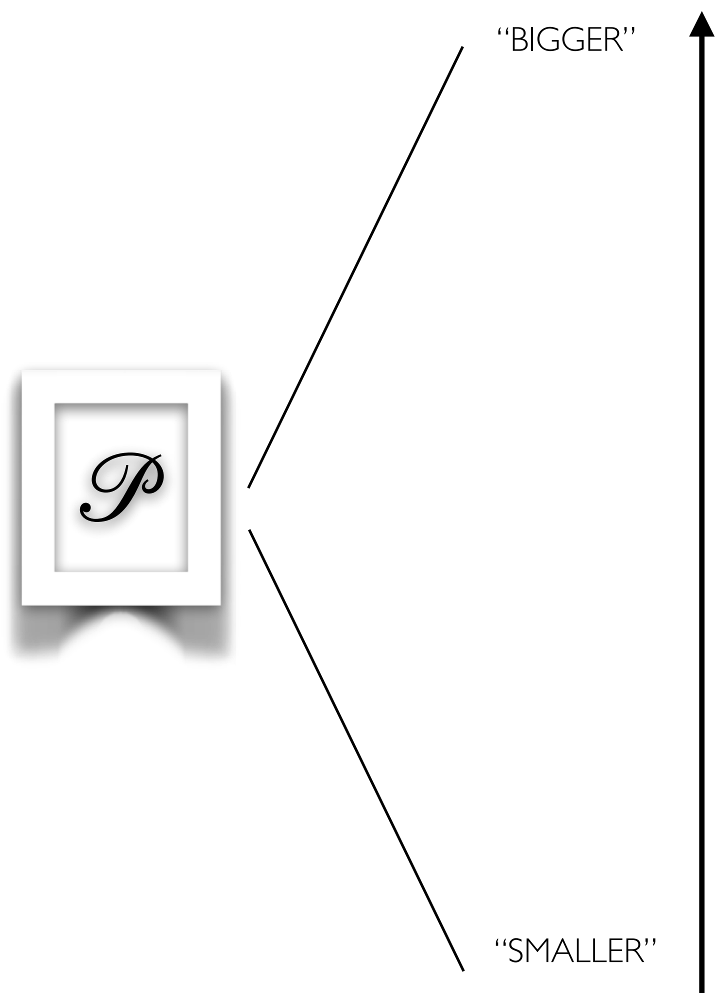
V Infinitely long programs that drive infinite super-τ machines^{-}.

IV Infinitely long programs that drive finite sub-τ machines^{-}.

III Finitely long programs that drive infinite super-τ machines^{-}.
(e.g. infinite-time Turing machines)

II Finitely long programs that drive infinite τ machines^{-}.
(e.g. Turing machines^{-}, register machines^{-})

I Finitely long programs that drive finite sub-τ machines^{-}.
(e.g. linear-bounded & finite-state automata)



V Infinitely long programs that drive infinite super-τ machines$^-$.

IV Infinitely long programs that drive finite sub-τ machines$^-$.

III Finitely long programs that drive infinite super-τ machines$^-$.

(e.g. infinite-time Turing machines)

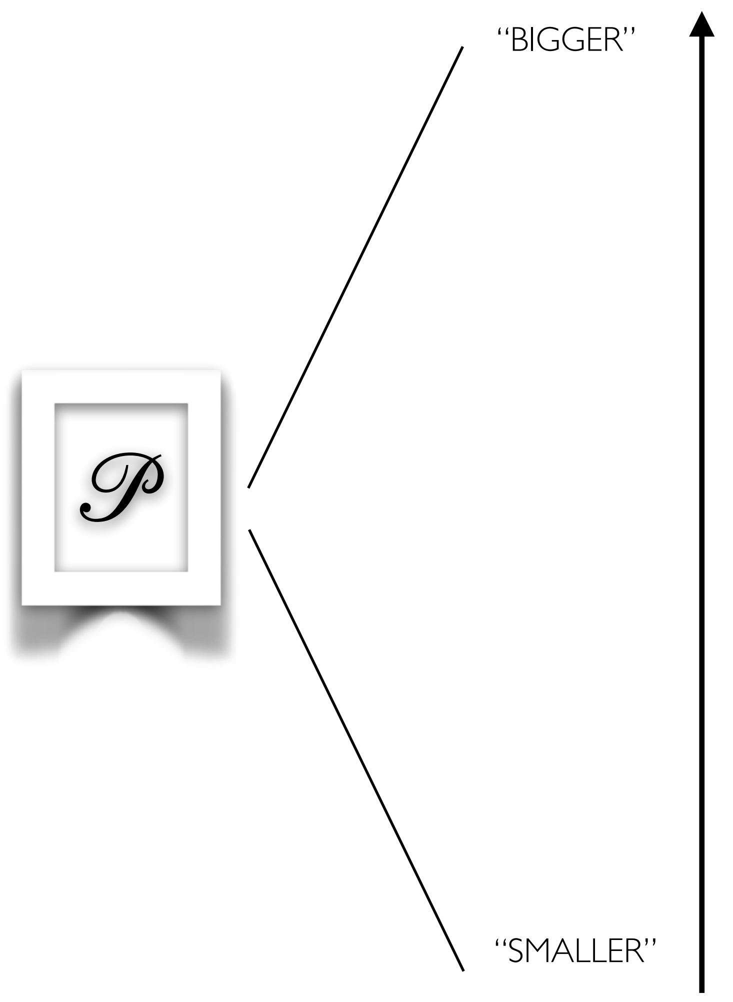
II Finitely long programs that drive infinite τ machines$^-$.

(e.g. Turing machines$^-$, register machines$^-$)

I Finitely long programs that drive finite sub-τ machines$^-$.

(e.g. linear-bounded & finite-state automata)

What is a τ machine?



V Infinitely long programs that drive infinite super-τ machines^{-}.

IV Infinitely long programs that drive finite sub-τ machines^{-}.

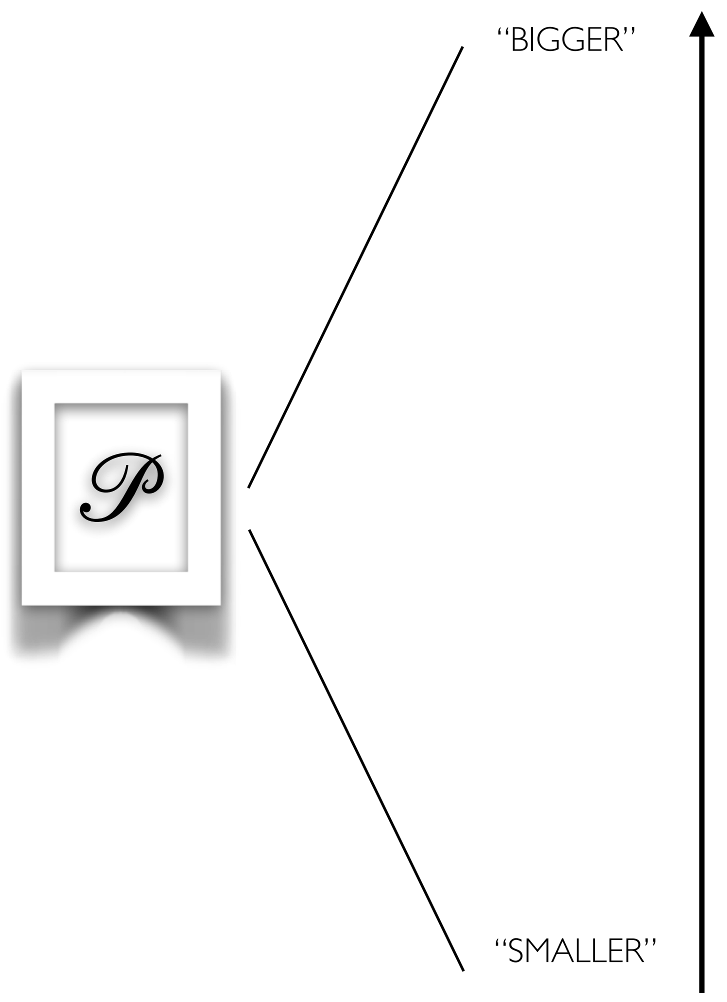
III Finitely long programs that drive infinite super-τ machines^{-}.
(e.g. infinite-time Turing machines)

II Finitely long programs that drive infinite τ machines^{-}.
(e.g. Turing machines^{-}, register machines^{-})

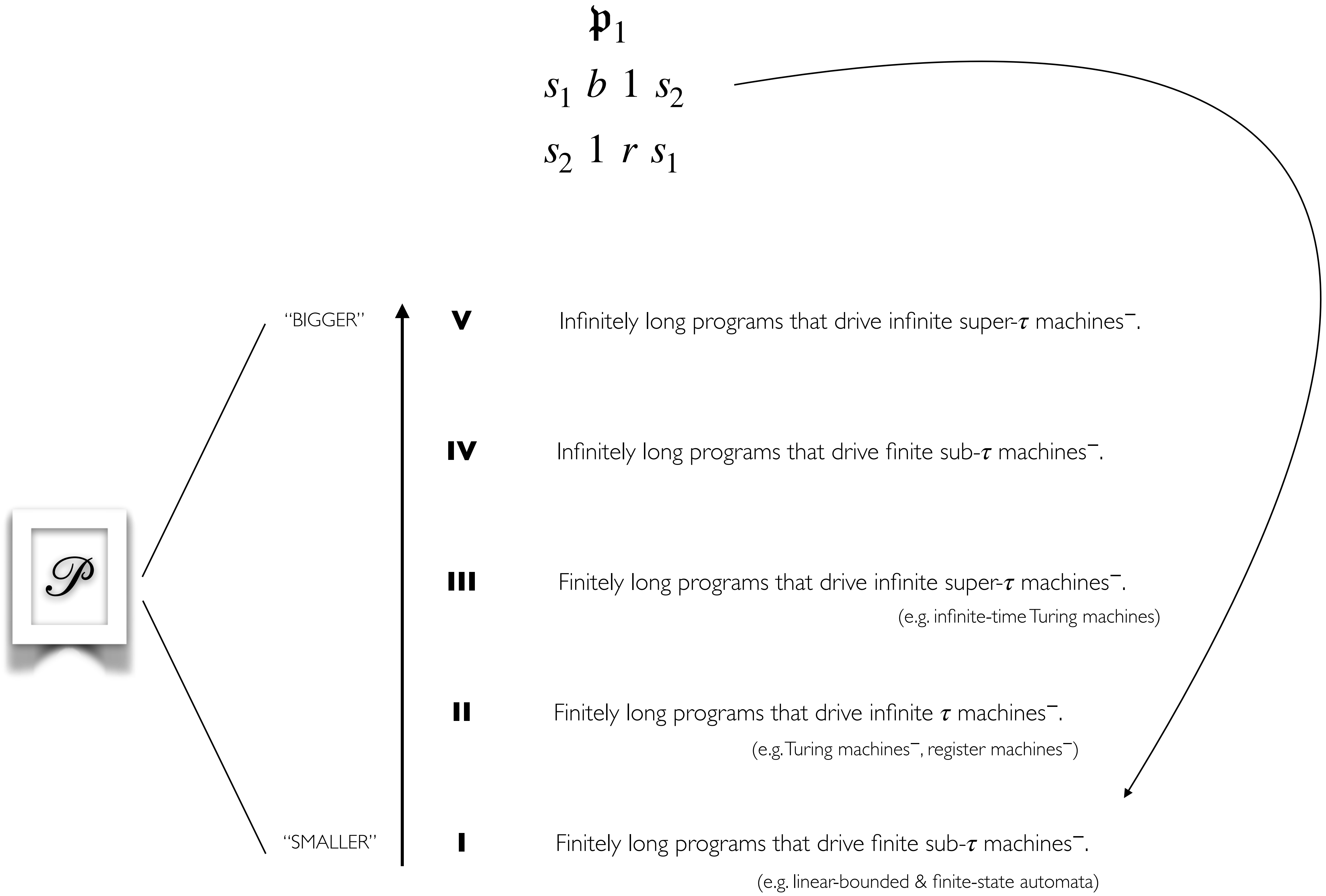
I Finitely long programs that drive finite sub-τ machines^{-}.
(e.g. linear-bounded & finite-state automata)

What is a τ machine?

What is the meaning of the superscript $^{-}$?



V	Infinitely long programs that drive infinite super-τ machines^{-}.
IV	Infinitely long programs that drive finite sub-τ machines^{-}.
III	Finitely long programs that drive infinite super-τ machines^{-}. (e.g. infinite-time Turing machines)
II	Finitely long programs that drive infinite τ machines^{-}. (e.g. Turing machines^{-}, register machines^{-})
I	Finitely long programs that drive finite sub-τ machines^{-}. (e.g. linear-bounded & finite-state automata)



$$\mathfrak{p}_1$$

$$s_1 \; b \; 1 \; s_2$$

$$s_2 \; 1 \; r \; s_1$$

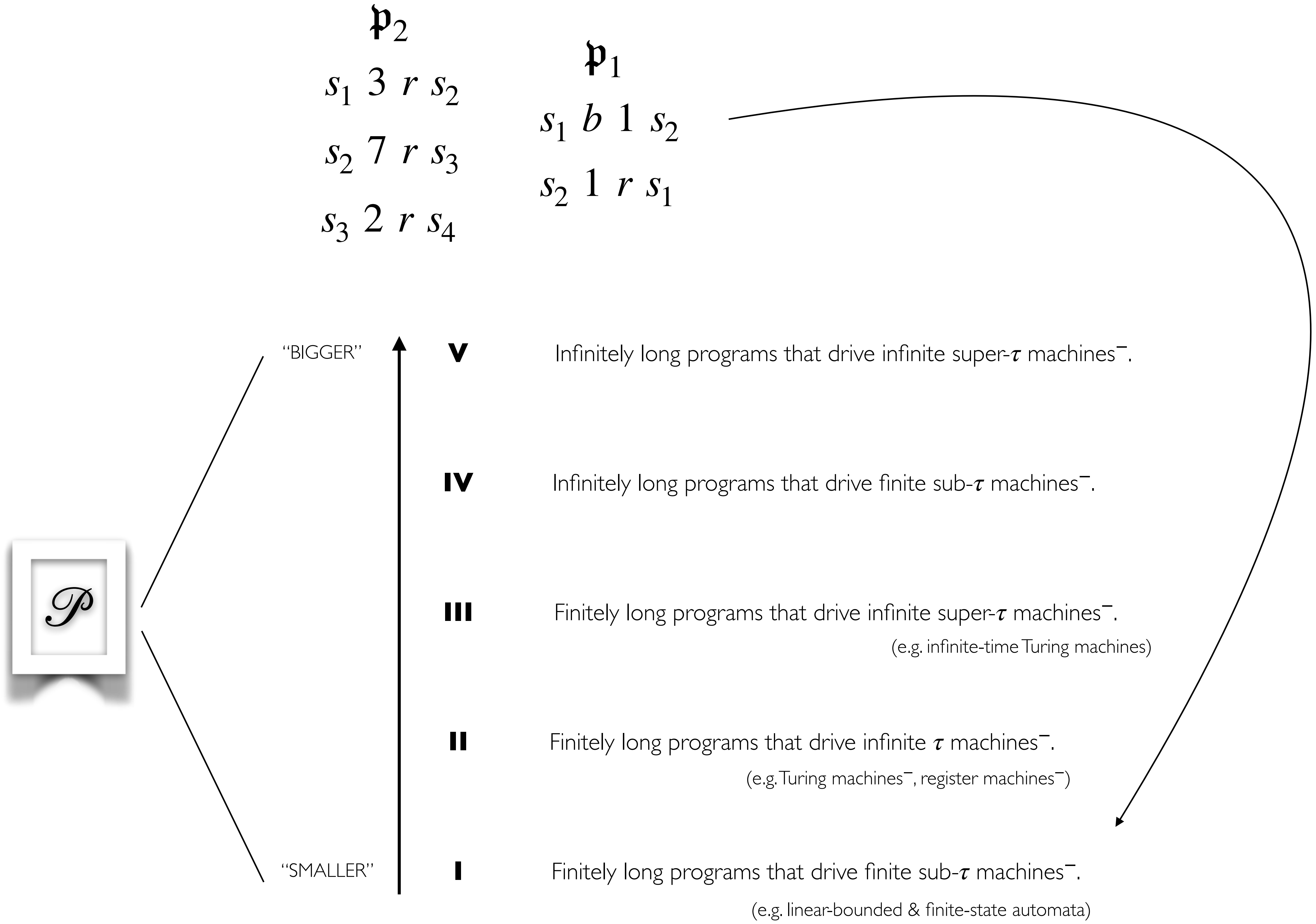
V Infinitely long programs that drive infinite super-τ machines^{-}.

IV Infinitely long programs that drive finite sub-τ machines^{-}.

III Finitely long programs that drive infinite super-τ machines^{-}.
(e.g. infinite-time Turing machines)

II Finitely long programs that drive infinite τ machines^{-}.
(e.g. Turing machines^{-}, register machines^{-})

I Finitely long programs that drive finite sub-τ machines^{-}.
(e.g. linear-bounded & finite-state automata)



$\mathfrak{p}_2$

$s_1\ 3\ r\ s_2$

$s_2\ 7\ r\ s_3$

$s_3\ 2\ r\ s_4$

$\mathfrak{p}_1$

$s_1\ b\ 1\ s_2$

$s_2\ 1\ r\ s_1$

$\mathscr{P}$

"BIGGER"

"SMALLER"

V Infinitely long programs that drive infinite super-τ machines$^-$.

IV Infinitely long programs that drive finite sub-τ machines$^-$.

III Finitely long programs that drive infinite super-τ machines$^-$.

(e.g. infinite-time Turing machines)

II Finitely long programs that drive infinite τ machines$^-$.

(e.g. Turing machines$^-$, register machines$^-$)

I Finitely long programs that drive finite sub-τ machines$^-$.

(e.g. linear-bounded & finite-state automata)

(In response to feedback in Bertinoro.)

$\mathfrak{p}_2$

$s_1\ 3\ r\ s_2$

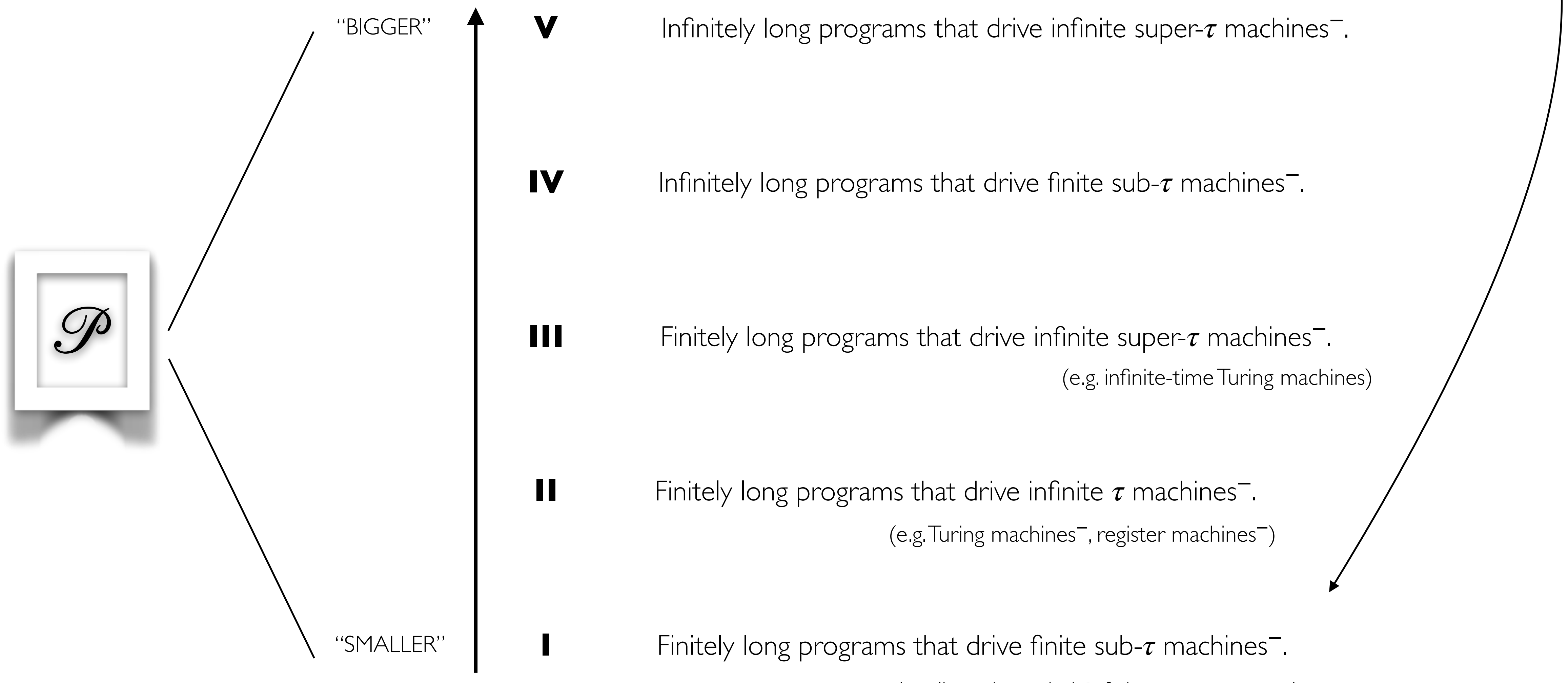
$s_2\ 7\ r\ s_3$

$s_3\ 2\ r\ s_4$

$\mathfrak{p}_1$

$s_1\ b\ 1\ s_2$

$s_2\ 1\ r\ s_1$



V Infinitely long programs that drive infinite super-τ machines$^-$.

IV Infinitely long programs that drive finite sub-τ machines$^-$.

III Finitely long programs that drive infinite super-τ machines$^-$.

(e.g. infinite-time Turing machines)

II Finitely long programs that drive infinite τ machines$^-$.

(e.g. Turing machines$^-$, register machines$^-$)

I Finitely long programs that drive finite sub-τ machines$^-$.

(e.g. linear-bounded & finite-state automata)

(In response to feedback in Bertinoro.)

$\mathfrak{p}_2$

s_1	3	r	s_2
s_2	7	r	s_3
s_3	2	r	s_4

$\mathfrak{p}_1$

s_1	b	1	s_2
s_2	1	r	s_1

$\mathscr{P}$

"BIGGER"

V Infinitely long programs that drive infinite super-τ machines$^-$.

IV Infinitely long programs that drive finite sub-τ machines$^-$.

III Finitely long programs that drive infinite super-τ machines$^-$.
(e.g. infinite-time Turing machines)

II Finitely long programs that drive infinite τ machines$^-$.
(e.g. Turing machines$^-$, register machines$^-$)

I Finitely long programs that drive finite sub-τ machines$^-$.
(e.g. linear-bounded & finite-state automata)

"SMALLER"

(In response to feedback in Bertinoro.)

$\mathfrak{p}_2$

$s_1\ 3\ r\ s_2$

$s_2\ 7\ r\ s_3$

$s_3\ 2\ r\ s_4$

$\mathfrak{p}_1$

$s_1\ b\ 1\ s_2$

$s_2\ 1\ r\ s_1$

"BIGGER"

V Infinitely long programs that drive infinite super-τ machines^{-}.

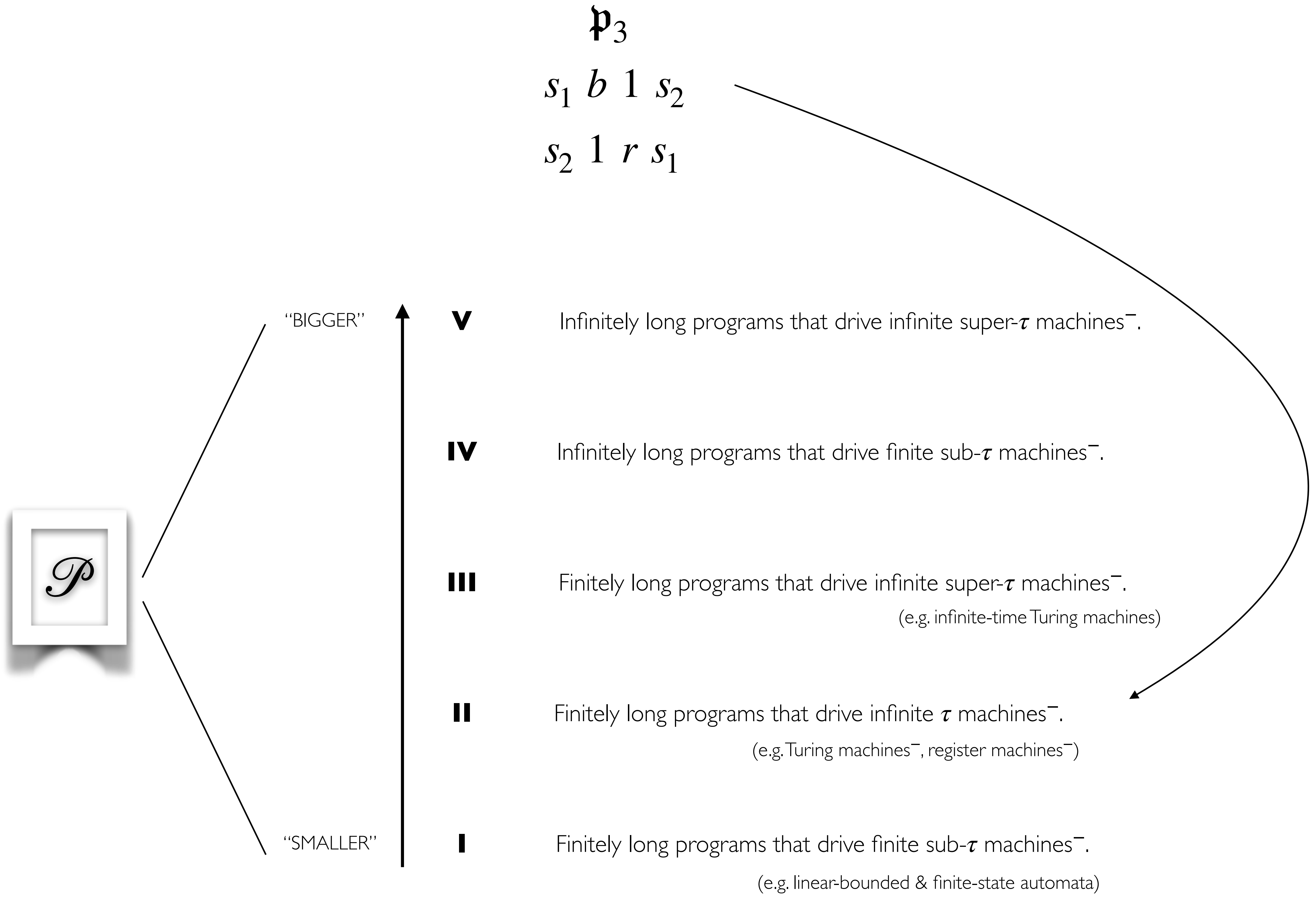
IV Infinitely long programs that drive finite sub-τ machines^{-}.

III Finitely long programs that drive infinite super-τ machines^{-}.
(e.g. infinite-time Turing machines)

II Finitely long programs that drive infinite τ machines^{-}.
(e.g. Turing machines^{-}, register machines^{-})

I Finitely long programs that drive finite sub-τ machines^{-}.
(e.g. linear-bounded & finite-state automata)

"SMALLER"



$\mathfrak{p}_3$

$s_1\ b\ 1\ s_2$

$s_2\ 1\ r\ s_1$

V Infinitely long programs that drive infinite super-τ machines$^-$.

IV Infinitely long programs that drive finite sub-τ machines$^-$.

III Finitely long programs that drive infinite super-τ machines$^-$.
(e.g. infinite-time Turing machines)

II Finitely long programs that drive infinite τ machines$^-$.
(e.g. Turing machines$^-$, register machines$^-$)

I Finitely long programs that drive finite sub-τ machines$^-$.
(e.g. linear-bounded & finite-state automata)

Theorem: The Halting Problem is Turing-unsolvable.

…

We assume an encoding of TMs that permits identification of each with some $m \in \mathbb{Z}^+$, and say that the binary halt function h maps a machine and its input to 1 if that machine halts, and to 2 if it doesn't:

$h(m, n) = 1$ if $m : n \longrightarrow$ halt
$h(m, n) = 2$ if $m : n \longrightarrow \infty$

So, the theorem we need can be expressed this way:

$(\star) \quad \neg \exists m^h \; [m^h$ computes $h]$

where a TM that computes a function *f* starts with arguments to *f* on its tape and goes to the value of *f* on applied to those arguments. Next, let's construct a TM m^c that copies a block of 1's (separated by a blank #), and (what BBJ in their *Computability & Logic* call) a "dithering" TM:

$m^d : n \longrightarrow$ halt if $n > 1$; $m^d : n \longrightarrow \infty$ if $n = 1$

We assume an encoding of TMs that permits identification of each with some $m \in \mathbb{Z}^+$, and say that the binary halt function h maps a machine and its input to 1 if that machine halts, and to 2 if it doesn't:

$$h(m,n) = 1 \text{ if } m : n \longrightarrow \text{ halt}$$
$$h(m,n) = 2 \text{ if } m : n \longrightarrow \infty$$

So, the theorem we need can be expressed this way:

$$(\star) \quad \neg\exists m^h \; [m^h \text{ computes } h]$$

where a TM that computes a function *f* starts with arguments to *f* on its tape and goes to the value of *f* on applied to those arguments. Next, let's construct a TM m^c that copies a block of 1's (separated by a blank #), and (what BBJ in their *Computability & Logic* call) a "dithering" TM:

$$m^d : n \longrightarrow \text{ halt if } n > 1; \;\; m^d : n \longrightarrow \infty \text{ if } n = 1$$

$\boxed{\mathfrak{p}_3'}$

Proof: Suppose for *reductio* that m^{h^*} [this is our witness for the existential quantifier in $(\star)$] computes h. Then we can make a composite machine m^3 consisting of m^c connected to and feeding m^{h^*} which is in turn connected to and feeding m^d. It's easy to see (use some paper and pencil/stylus and tablet!) that

$$(1) \quad \text{if } h(n,n) = 1, \text{ then } m^3 : n \longrightarrow \infty$$

and

$$(2) \quad \text{if } h(n,n) = 2, \text{ then } m^3 : n \longrightarrow \text{ halt}\,.$$

To reach our desired contradiction, we simply ask: What happens when we instantiate n to m^3 in (1) and (2)? (E.g., perhaps the TM m^3 is 5, then we would have $h(5{,}5)$.) The answer to this question, and its leading directly to just what the doctor ordered, is left to the reader. **QED**

“$\mathfrak{p}_4$”

Print ‘non-halter’

Simulate the input (encoded) TM

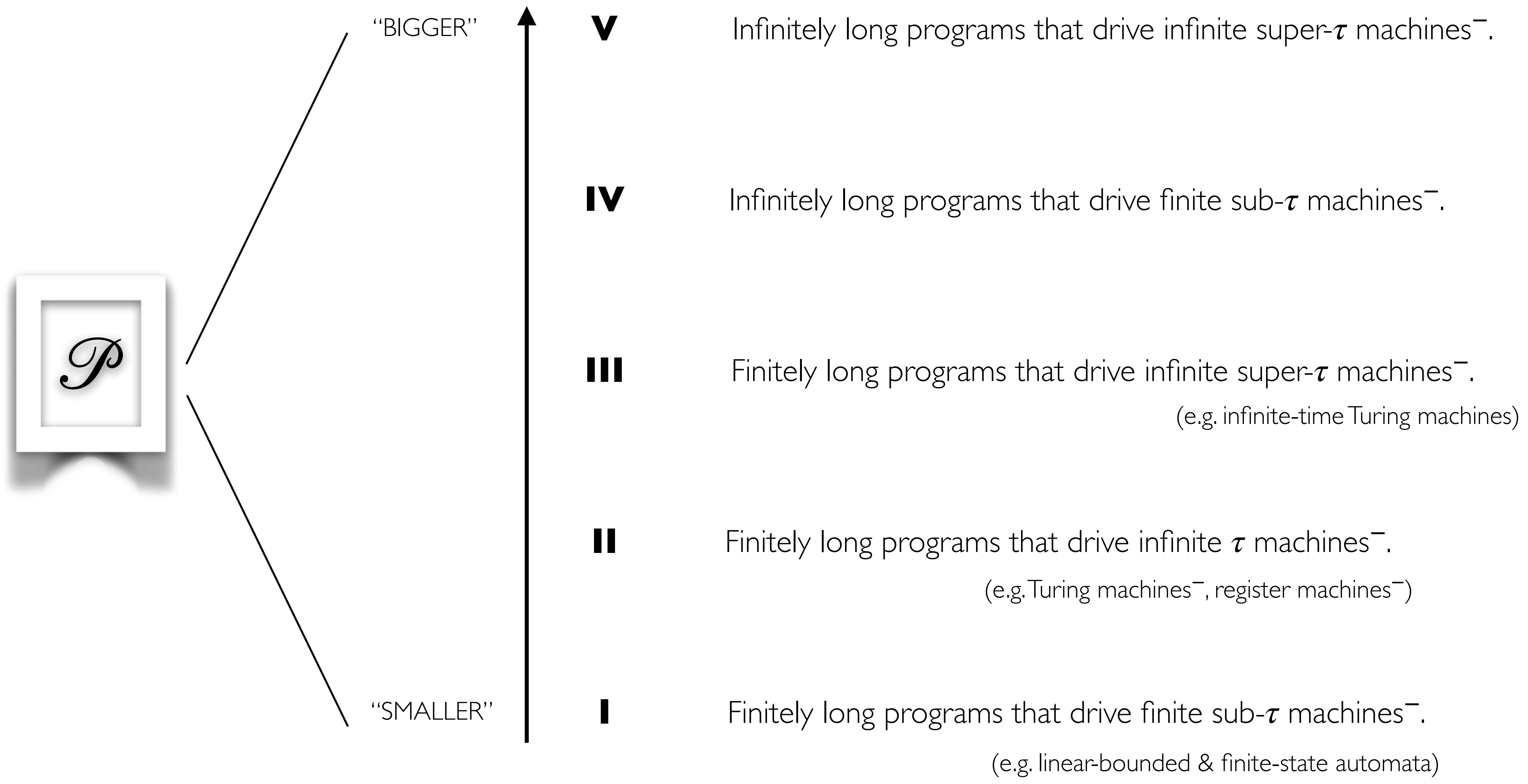
If the simulation halts, print ‘halter’

“$\mathfrak{p}_4$”

```
Print ‘non-halter’

Simulate the input (encoded) TM

If the simulation halts, print ‘halter’
```



V Infinitely long programs that drive infinite super-τ machines^{-}.

IV Infinitely long programs that drive finite sub-τ machines^{-}.

III Finitely long programs that drive infinite super-τ machines^{-}.
(e.g. infinite-time Turing machines)

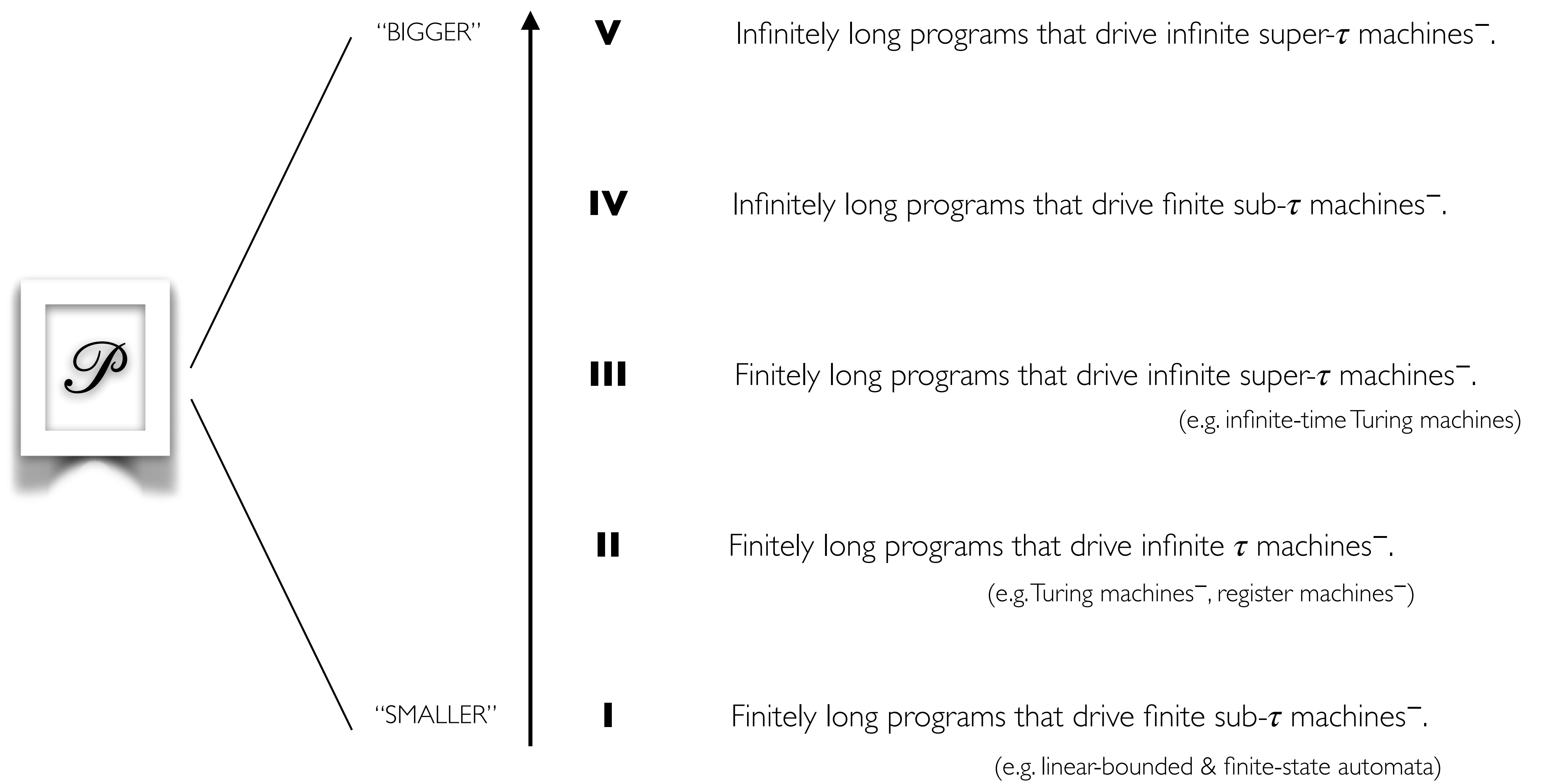
II Finitely long programs that drive infinite τ machines^{-}.
(e.g. Turing machines^{-}, register machines^{-})

I Finitely long programs that drive finite sub-τ machines^{-}.
(e.g. linear-bounded & finite-state automata)

$\mathfrak{p}_5$

[`M Davis imperative program with oracle calls`]

```
$L_1$   $X \longleftarrow X - 1$
$L_2$   $Y \longleftarrow Y + 1$
$L_3$   IF $X \neq 0$ GOTO $L_1$
```



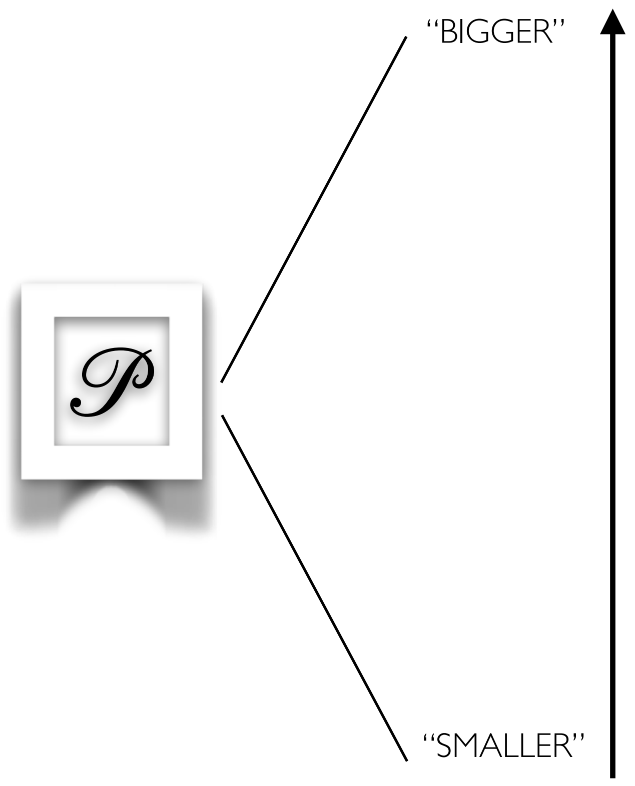
V Infinitely long programs that drive infinite super-τ machines^{-}.

IV Infinitely long programs that drive finite sub-τ machines^{-}.

III Finitely long programs that drive infinite super-τ machines^{-}.
(e.g. infinite-time Turing machines)

II Finitely long programs that drive infinite τ machines^{-}.
(e.g. Turing machines^{-}, register machines^{-})

I Finitely long programs that drive finite sub-τ machines^{-}.
(e.g. linear-bounded & finite-state automata)



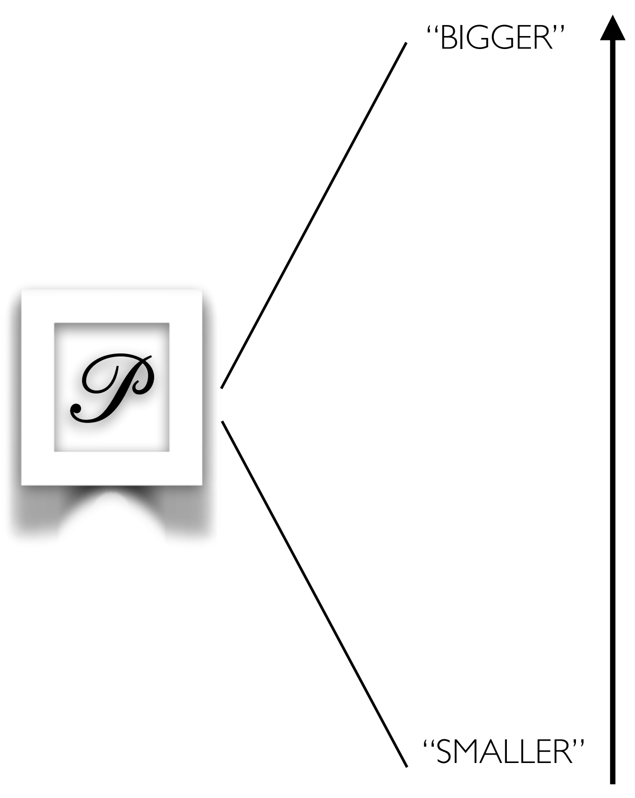
V Infinitely long programs that drive infinite super-τ machines^{-}.

IV Infinitely long programs that drive finite sub-τ machines^{-}.

III Finitely long programs that drive infinite super-τ machines^{-}.
(e.g. infinite-time Turing machines)

II Finitely long programs that drive infinite τ machines^{-}.
(e.g. Turing machines^{-}, register machines^{-})

I Finitely long programs that drive finite sub-τ machines^{-}.
(e.g. linear-bounded & finite-state automata)



V Infinitely long programs that drive infinite super-τ machines^{-}. ?

IV Infinitely long programs that drive finite sub-τ machines^{-}.

III Finitely long programs that drive infinite super-τ machines^{-}.
(e.g. infinite-time Turing machines)

II Finitely long programs that drive infinite τ machines^{-}.
(e.g. Turing machines^{-}, register machines^{-})

I Finitely long programs that drive finite sub-τ machines^{-}.
(e.g. linear-bounded & finite-state automata)

(Pure General) Logic Programming …

There are *Two* Logicist Branches; B1:

There are *Two* Logicist Branches; B1:

Frege, 1893:
“Aha! Currying! I recast multiple-arity operations with functions into a unary affair!”

There are *Two* Logicist Branches; B1:

Frege, 1893:
“Aha! Currying! I recast multiple-arity operations with functions into a unary affair!”

Schönfinkel, 1920’s:
“Aha! I can do this stuff using combinatory logic!”

There are *Two* Logicist Branches; B1:

Frege, 1893:
“Aha! Currying! I recast multiple-arity operations with functions into a unary affair!”

Schönfinkel, 1920’s:
“Aha! I can do this stuff using combinatory logic!”

Church, 1920’s & 30’s:
“Aha! The lambda calculus!

There are *Two* Logicist Branches; B1:

Frege, 1893:
"Aha! Currying! I recast multiple-arity operations with functions into a unary affair!"

Schönfinkel, 1920's:
"Aha! I can do this stuff using combinatory logic!"

Church, 1920's & 30's:
"Aha! The lambda calculus!

...

There are *Two* Logicist Branches; B1:

Frege, 1893:
"Aha! Currying! I recast multiple-arity operations with functions into a unary affair!"

Schönfinkel, 1920's:
"Aha! I can do this stuff using combinatory logic!"

Church, 1920's & 30's:
"Aha! The lambda calculus!

...

Haskell

There are *Two* Logicist Branches; B1:

Frege, 1893:
"Aha! Currying! I recast multiple-arity operations with functions into a unary affair!"

Schönfinkel, 1920's:
"Aha! I can do this stuff using combinatory logic!"

Church, 1920's & 30's:
"Aha! The lambda calculus!

...

<u>Haskell</u> OCaml, Scheme, …

There are *Two* Logicist Branches; B1:

Frege, 1893:
"Aha! Currying! I recast multiple-arity operations with functions into a unary affair!"

Schönfinkel, 1920's:
"Aha! I can do this stuff using combinatory logic!"

Church, 1920's & 30's:
"Aha! The lambda calculus!

…

Haskell OCaml, Scheme, …
Athena

Two Logicist Branches; B2:

Two Logicist Branches; B2:

The AI Branch: Automated Reasoning

Two Logicist Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

Two Logicist Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

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

Two Logicist Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

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

...

Two Logicist Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

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

…

Prolog?

Two Logicist Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

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

...

Two Logicist Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

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

...

PGLP

Two Logicist Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

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

…

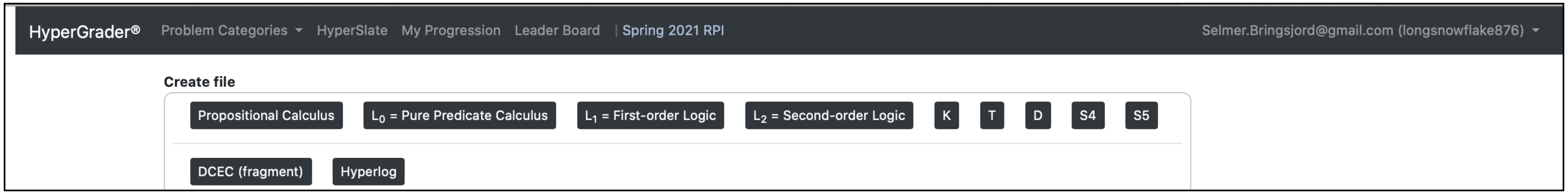
PGLP

Two Logicist Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

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



PGLP

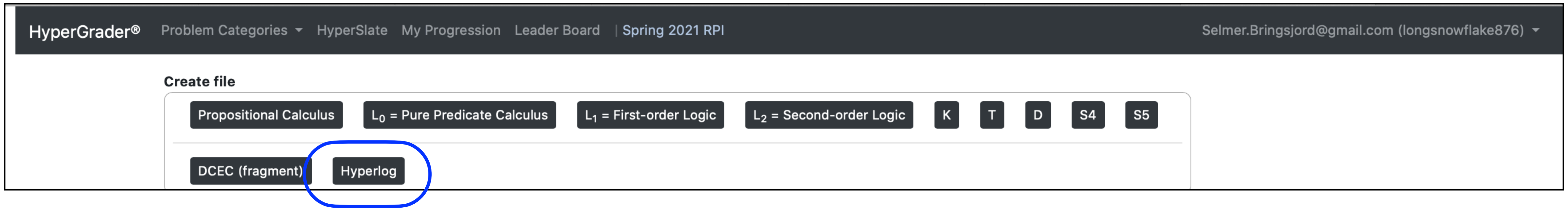
HyperSlate® : HyperLog

Two Logicist Branches; B2:

The AI Branch: Automated Reasoning

Leibniz

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



PGLP

HyperSlate® : HyperLog

HyperLog: Historico-logico-programming Landscape

First "logic programs"
300 BC

Liebniiz
Dies 1716

Frege
1893

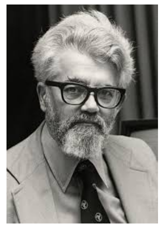
Schöenfinkel
1893

Combinatory Logic

simple type theory

Church

λ-calculus


Logic Theorist
(birth of modern logicist AI)

Simon

1956

Turing

Lisp

Prolog

Fortran

Athena

ML

Scheme

CL

Lisp Family

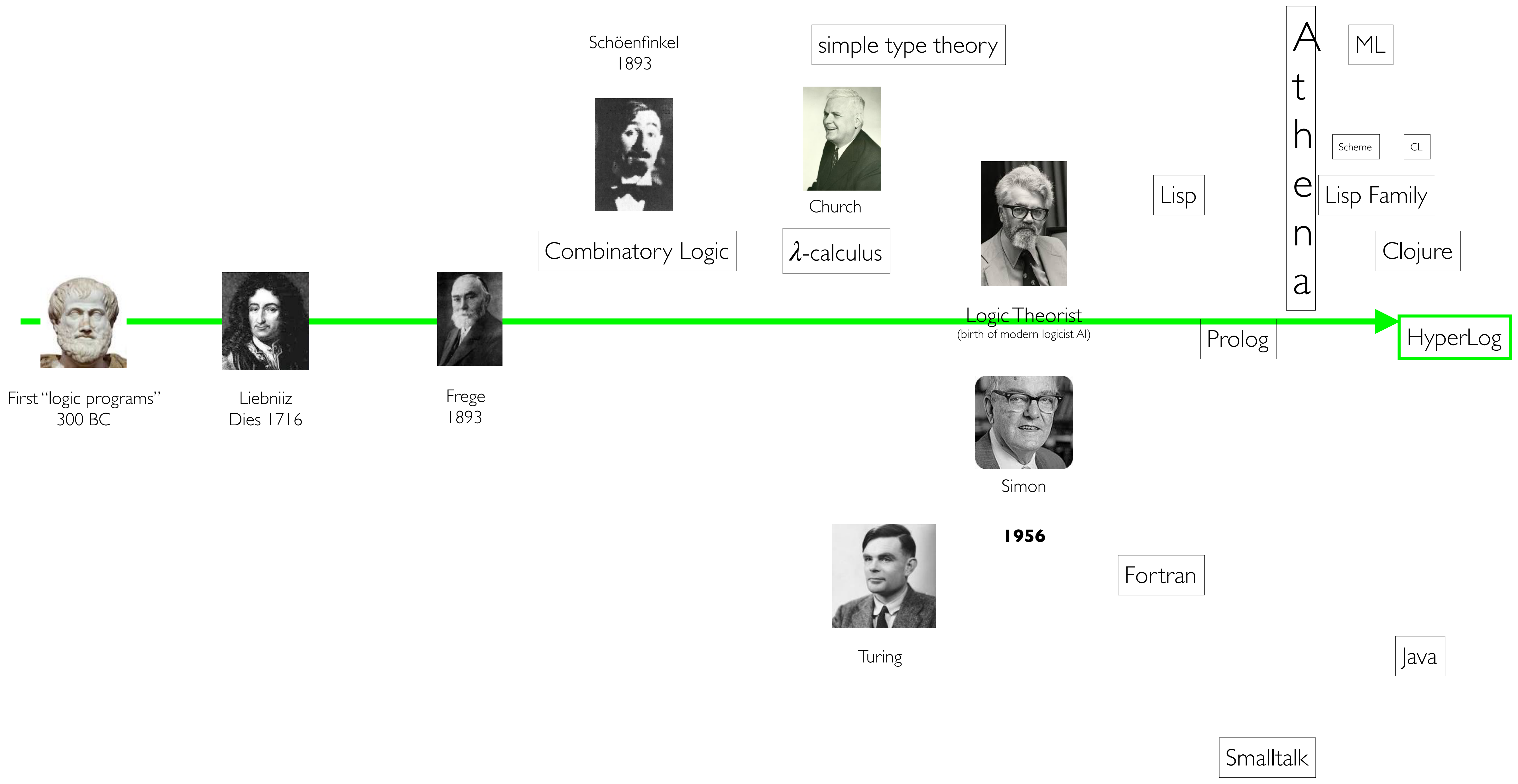
Clojure

HyperLog

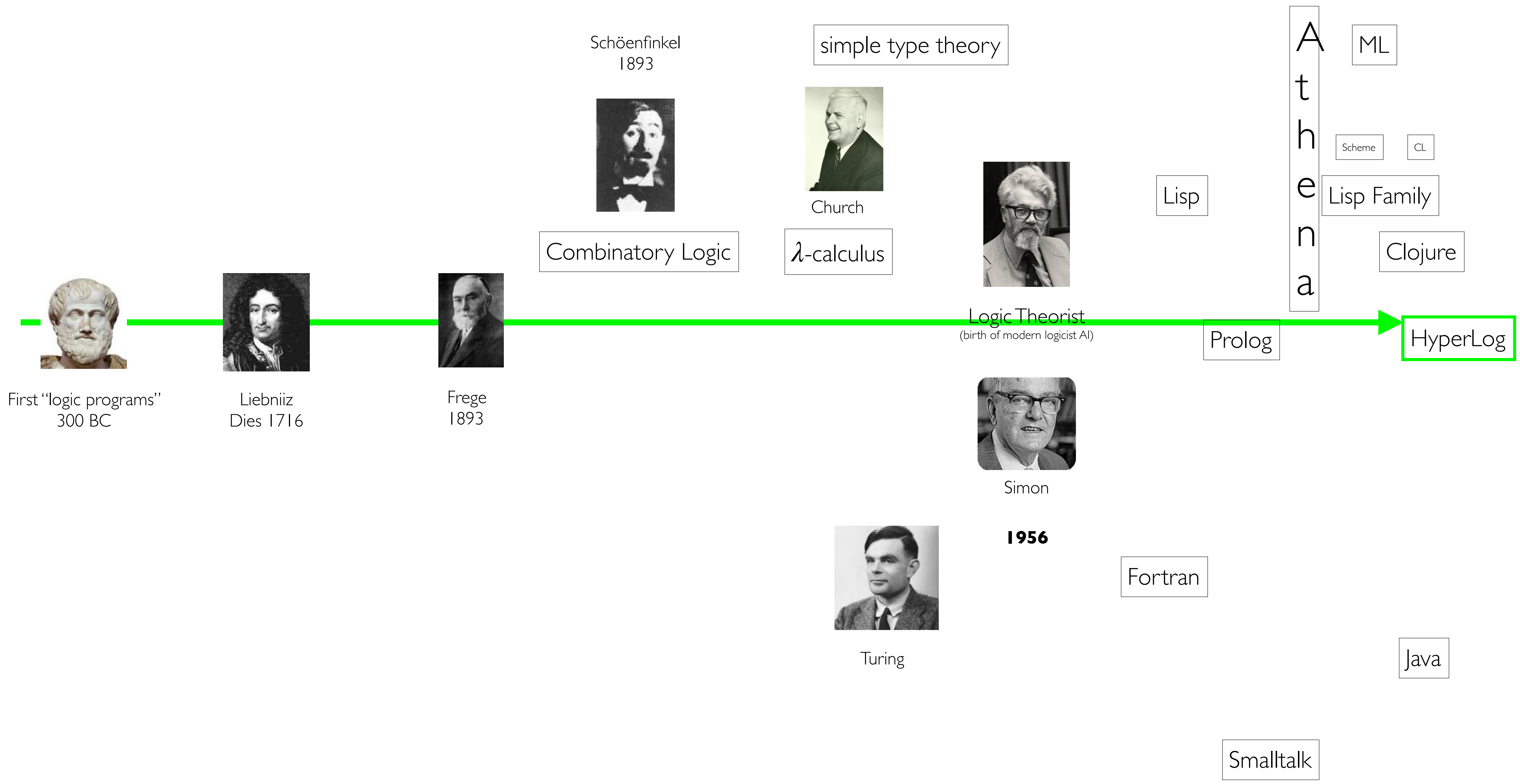
Java

Smalltalk

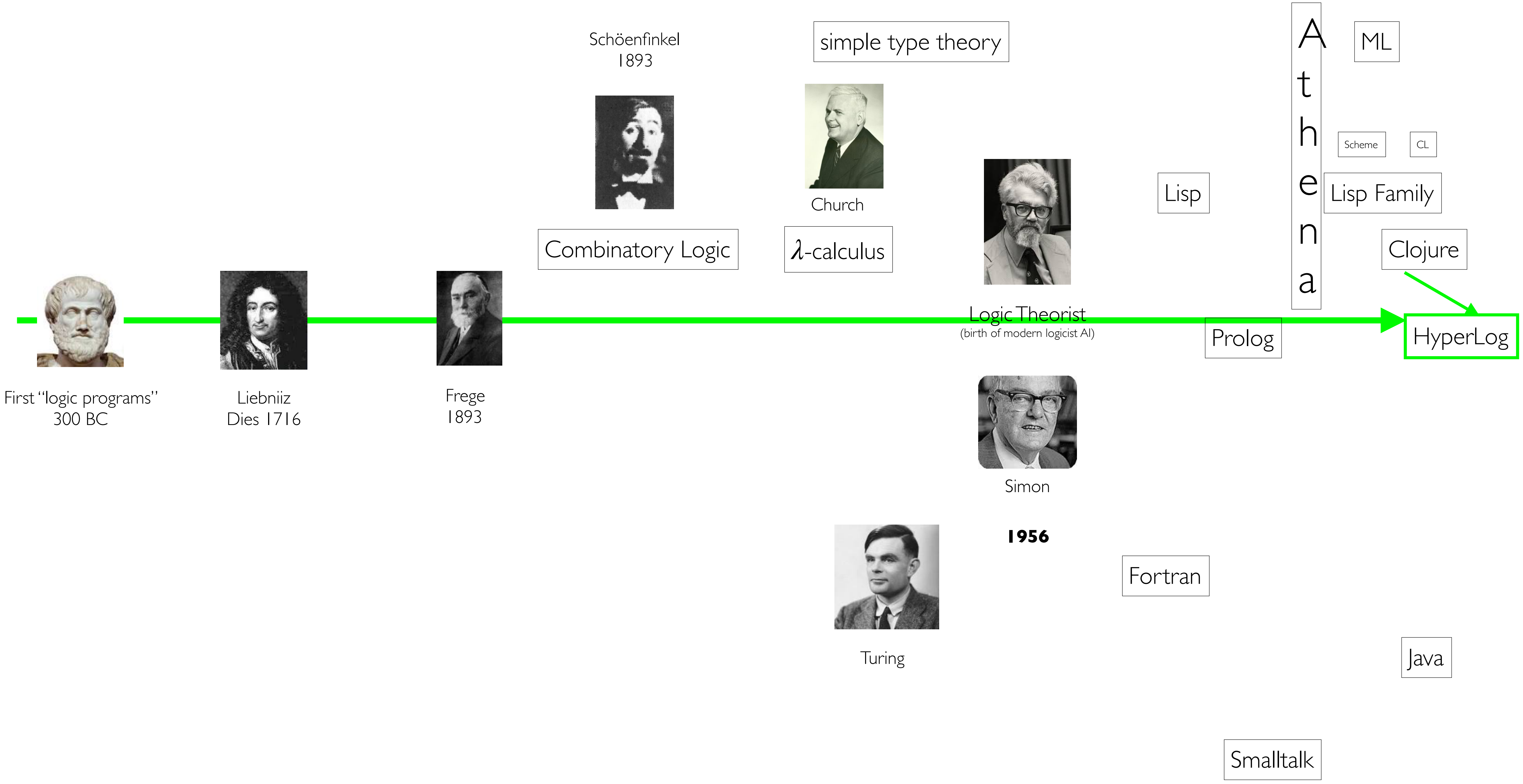
HyperLog: Historico-logico-programming Landscape



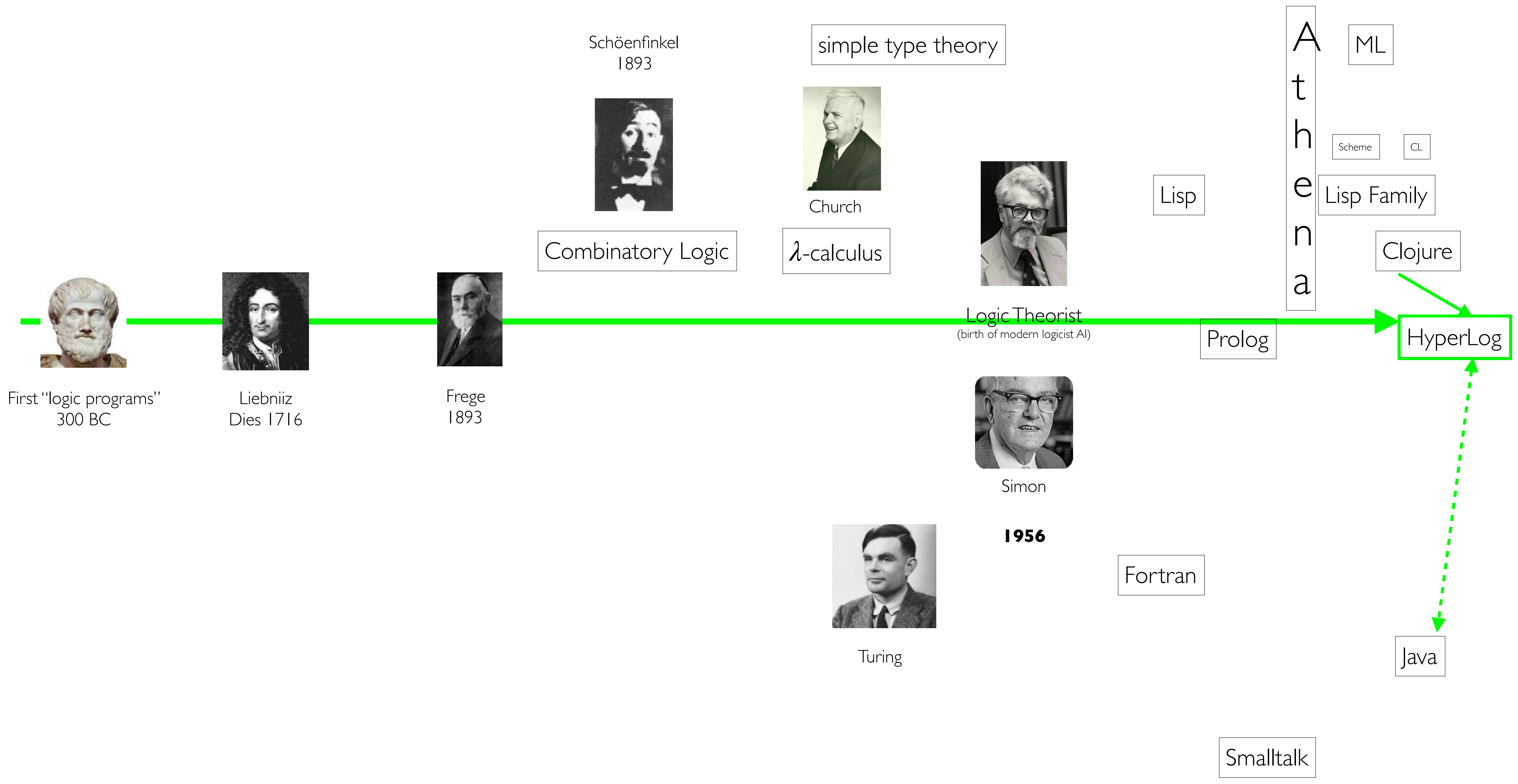
HyperLog: Historico-logico-programming Landscape



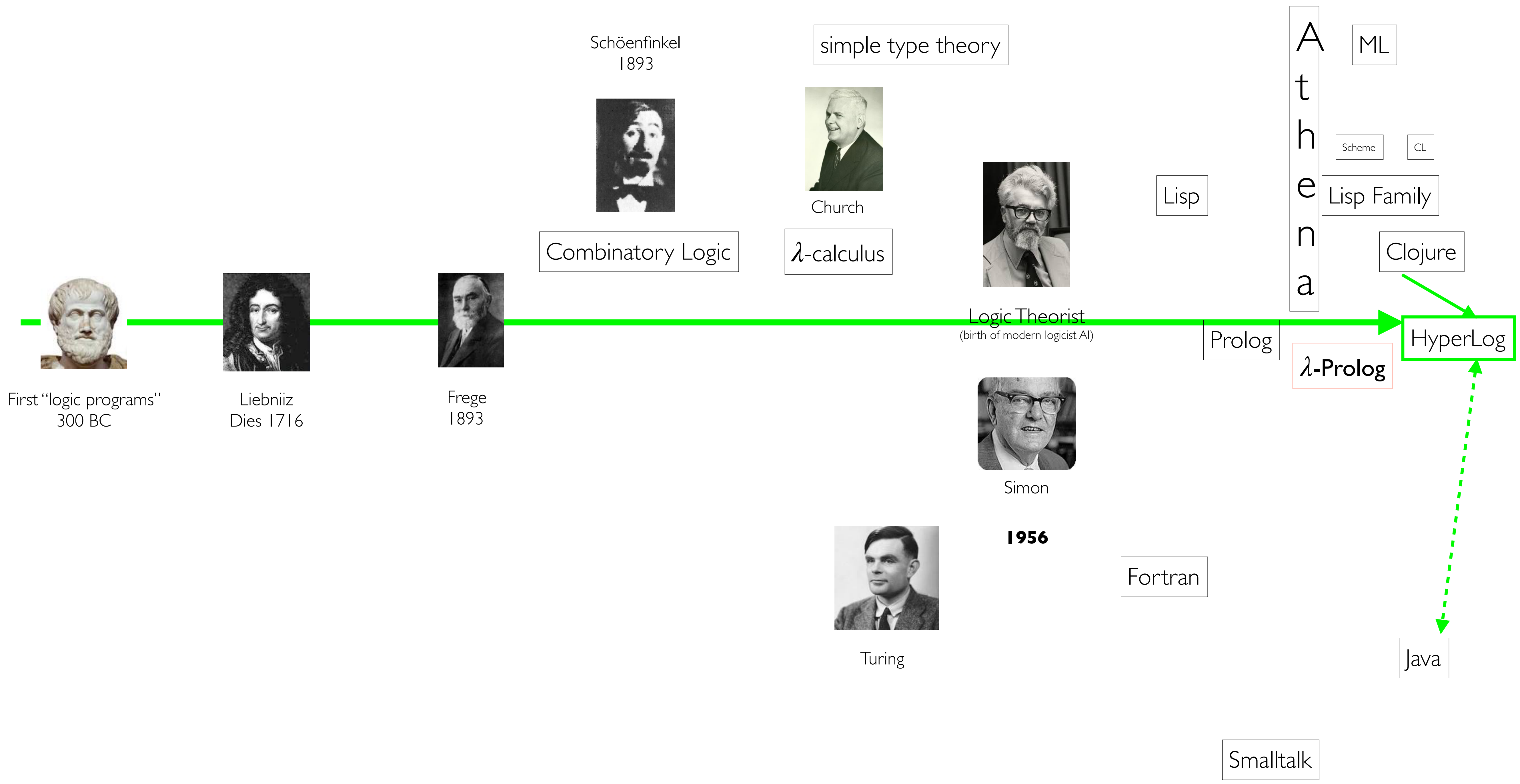
HyperLog: Historico-logico-programming Landscape



HyperLog: Historico-logico-programming Landscape



HyperLog: Historico-logico-programming Landscape



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

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

degree of "confidence"

program

proof(s)

query

argument(s)

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

reasoner

checker

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.

program

query

degree of "confidence"

proof(s)

argument(s)

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

reasoner

checker

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.

program

degree of "confidence"

proof(s)

query

argument(s)

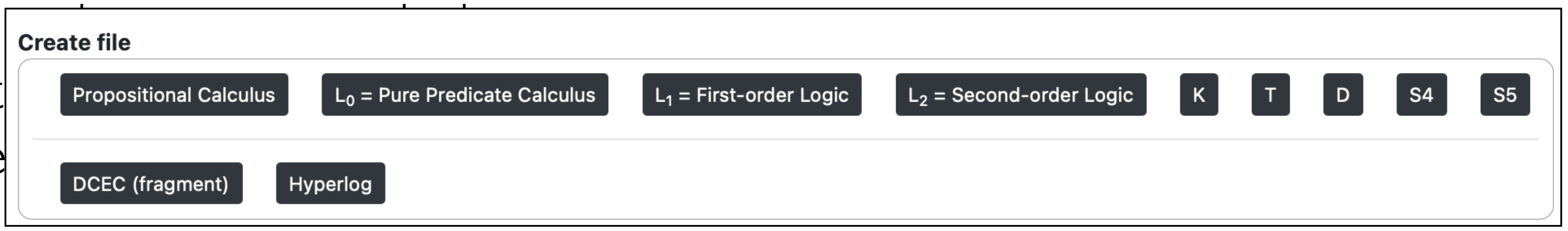
$$\mathscr{L} = \langle \mathcal{L}, \mathcal{I} \rangle$$

$$\begin{array}{c} \mathbb{P} \quad \boldsymbol{\mathcal{L}} \\ \mathfrak{q} \quad \boldsymbol{\mathcal{L}} \\ \hline \mathbb{R} \; : \; \langle \mathbb{P}, \mathfrak{q} \rangle \longrightarrow \langle \texttt{Y} | \texttt{N} | \texttt{U}, \delta, \pi_{(s)} | \alpha_{(s)} \rangle \\ \mathbb{C} \; : \; \pi_{(s)} | \alpha_{(s)} \longrightarrow \langle \texttt{Y} | \texttt{N} | \texttt{U}, \delta \rangle \end{array}$$

reasoner

checker

For just "logic programming," and
goes back to circa 1970, restrict t
thereof, and use resolution as the


Create file
Propositional Calculus
L_0 = Pure Predicate Calculus
L_1 = First-order Logic
L_2 = Second-order Logic
K
T
D
S4
S5
DCEC (fragment)
Hyperlog

program

degree of "confidence"

proof(s)

query

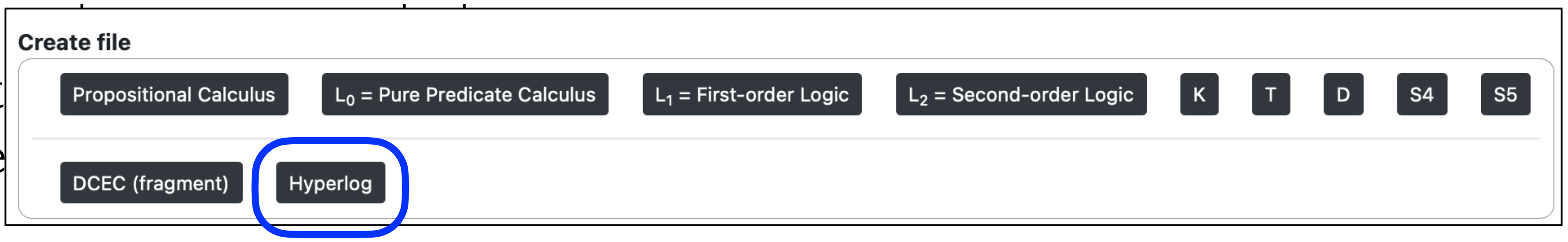
argument(s)

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

reasoner

checker

For just "logic programming," and
goes back to circa 1970, restrict t
thereof, and use resolution as the


Create file
Propositional Calculus
L_0 = Pure Predicate Calculus
L_1 = First-order Logic
L_2 = Second-order Logic
K
T
D
S4
S5
DCEC (fragment)
[✓ Hyperlog]

program

query

degree of "confidence"

proof(s)

argument(s)

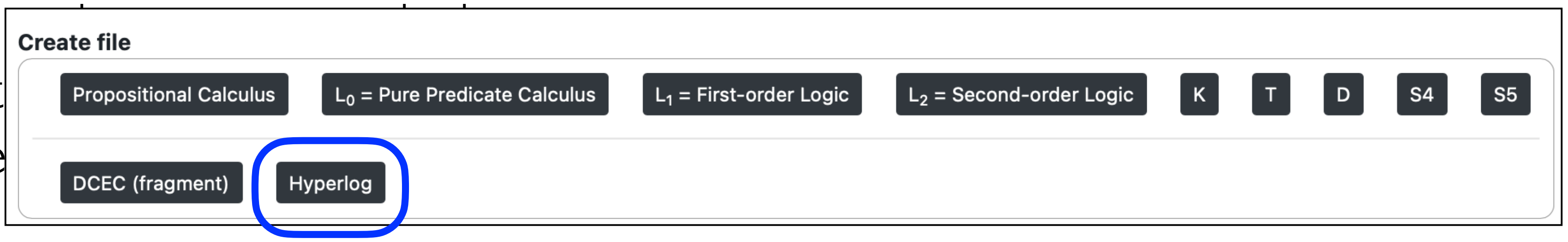
$$\mathscr{L} = \langle \mathcal{L}, \mathcal{I} \rangle$$

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

reasoner

checker

For just "logic programming," and
goes back to circa 1970, restrict t
thereof, and use resolution as the


Create file
Propositional Calculus
L_0 = Pure Predicate Calculus
L_1 = First-order Logic
L_2 = Second-order Logic
K
T
D
S4
S5
DCEC (fragment)
Hyperlog

"Live" function symbols in here allowed.

program

degree of "confidence"

proof(s)

query

argument(s)

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

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

reasoner

checker

On the Anatomy of a PGLP Program

On the Anatomy of a PGLP Program

Linguistics

$\vdots$	$\vdots$	$\vdots$
L_2^μ	meta-level$_2$ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level$_1$ language	$\exists x \ \mathrm{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{A} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

On the Anatomy of a PGLP Program

Linguistics

$\vdots$	$\vdots$	$\vdots$
L_2^μ	meta-level$_2$ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level$_1$ language	$\exists x \ \mathrm{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{U} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of (deductive and/or inductive) inference schemata.

On the Anatomy of a PGLP Program

Linguistics

$\vdots$	$\vdots$	$\vdots$
L_2^μ	meta-level$_2$ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level$_1$ language	$\exists x \ \mathrm{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{U} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of (deductive and/or inductive) inference schemata.

Semantics

Reasoning-semantic; wholly inferentialist (after all, what's the semantics of deduction over meta-level$_1$ formulae??).

On the Anatomy of a PGLP Program

Linguistics

$\vdots \qquad \vdots \qquad \vdots$

L_2^μ	meta-level$_2$ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level$_1$ language	$\exists x \ \mathrm{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{A} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of (deductive and/or inductive) inference schemata.

Semantics

Reasoning-semantic; wholly inferentialist (after all, what's the semantics of deduction over meta-level$_1$ formulae??).

On the Anatomy of a PGLP Program

Linguistics

$$\vdots \qquad \vdots \qquad \vdots$$

L_2^μ	meta-level$_2$ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level$_1$ language	$\exists x \ \mathrm{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{U} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of (deductive and/or inductive) inference schemata.

Semantics

Reasoning-semantic; wholly inferentialist (after all, what's the semantics of deduction over meta-level$_1$ formulae??).

$$\mathscr{L}$$

On the Anatomy of a PGLP Program

Linguistics

$\vdots$	$\vdots$	$\vdots$
L_2^μ	meta-level2 language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level1 language	$\exists x \ \mathrm{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{U} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of (deductive and/or inductive) inference schemata.

Semantics

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

$\mathscr{L}$

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

On the Anatomy of a PGLP Program

Linguistics

$\vdots \quad \vdots \quad \vdots$

L_2^μ meta-level₂ language $(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$

L_1^μ meta-level₁ language $\exists x \ \mathrm{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{U} \models \phi$

L object-level language $\phi \quad \psi \quad \delta$

Inference

A collection of (deductive and/or inductive) inference schemata.

Semantics

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

$$\mathscr{L}$$

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

On the Anatomy of a PGLP Program

Linguistics

$\vdots$	$\vdots$	$\vdots$
L_2^μ	meta-level₂ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level₁ language	$\exists x \ \mathbf{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{U} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of (deductive and/or inductive) inference schemata.

Semantics

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

$\mathscr{L}$

+ ShadowReasoner

On the Anatomy of a PGLP Program

Linguistics

$\vdots$	$\vdots$	$\vdots$
L_2^μ	meta-level₂ language	$(\{\phi\} \vdash \psi \wedge \{\psi\} \vdash \delta) \vdash_{\mu_2} \{\phi\} \vdash \delta$
L_1^μ	meta-level₁ language	$\exists x \ \mathrm{rank}(\phi) = x \quad \{\phi\} \vdash \psi \quad \mathfrak{A} \models \phi$
L	object-level language	$\phi \quad \psi \quad \delta$

Inference

A collection of (deductive and/or inductive) inference schemata.

Semantics

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

$$\mathscr{L}$$

$$\mathfrak{p}_5$$

$$\phi_1 \vee \phi_2 \vee \phi_3 \vee \ldots$$

$$\neg\phi_2 \wedge \neg\phi_3 \wedge \neg\phi_4 \wedge \ldots$$

$$q = \phi_1$$

$\mathscr{P}$

"BIGGER"

V Infinitely long programs that drive infinite super-τ machines^{-}.

IV Infinitely long programs that drive finite sub-τ machines^{-}.

III Finitely long programs that drive infinite super-τ machines^{-}.
(e.g. infinite-time Turing machines)

II Finitely long programs that drive infinite τ machines^{-}.
(e.g. Turing machines^{-}, register machines^{-})

I Finitely long programs that drive finite sub-τ machines^{-}.
(e.g. linear-bounded & finite-state automata)

"SMALLER"

Concerns/Objections …

Concerns/Objections

Concerns/Objections

- "Is $\mathscr{P}$ consistent with Post's comments regarding 1-systems and a 'natural law'?"

Concerns/Objections

- “Is $\mathscr{P}$ consistent with Post’s comments regarding 1-systems and a ‘natural law’?”
- “What about grammars?”

Concerns/Objections

- "Is $\mathscr{P}$ consistent with Post's comments regarding 1-systems and a 'natural law'?"
- "What about grammars?"
- "But doesn't Blum's Size Theorem/Speedup Theorem tell us that we already well understood the concept of the size of a program, and that understanding runs counter to yours?"

Concerns/Objections

- "Is $\mathscr{P}$ consistent with Post's comments regarding 1-systems and a 'natural law'?"
- "What about grammars?"
- "But doesn't Blum's Size Theorem/Speedup Theorem tell us that we already well understood the concept of the size of a program, and that understanding runs counter to yours?"
  - Actually, *Gödel's* Speedup Theorem is a good match for the point of view that has been set out here (since speedup comes from moving to second-, third-, …, n-, n+1-order … logic).

Loggik kan hjelpe deg å leve for alltid.

Church’s Theorem & its proof …

Church's Theorem: The *Entscheidungsproblem* is Turing-unsolvable.

Proof-sketch: We need to show that the question $\Phi \vdash \phi$? is not Turing-decidable. (Here we are working within the framework of $\mathscr{L}_1$.) To begin, note that competent users of HyperSlate® know that any Turing machine m can be formalized in a HyperSlate® workspace. (Explore! Prove it to yourself in hands-on fashion!) They will also then know that

$$(\dagger) \quad \forall m, n \in \mathbb{N}\ \exists \Phi, \phi\ [\Phi \vdash \phi \leftrightarrow\ m : n \longrightarrow \text{ halt}]$$

where Φ and ϕ are built in HyperSlate®.

Now, let's assume for contradiction that theoremhood in first-order logic can be decided by a Turing machine m_t. But this is absurd. Why? Because imagine that someone now comes to us asking whether some arbitrary TM m halts. We can infallibly and algorithmically supply a correct answer, because we can formalize m in line with (†) and then employ m_t to given us the answer. **QED**