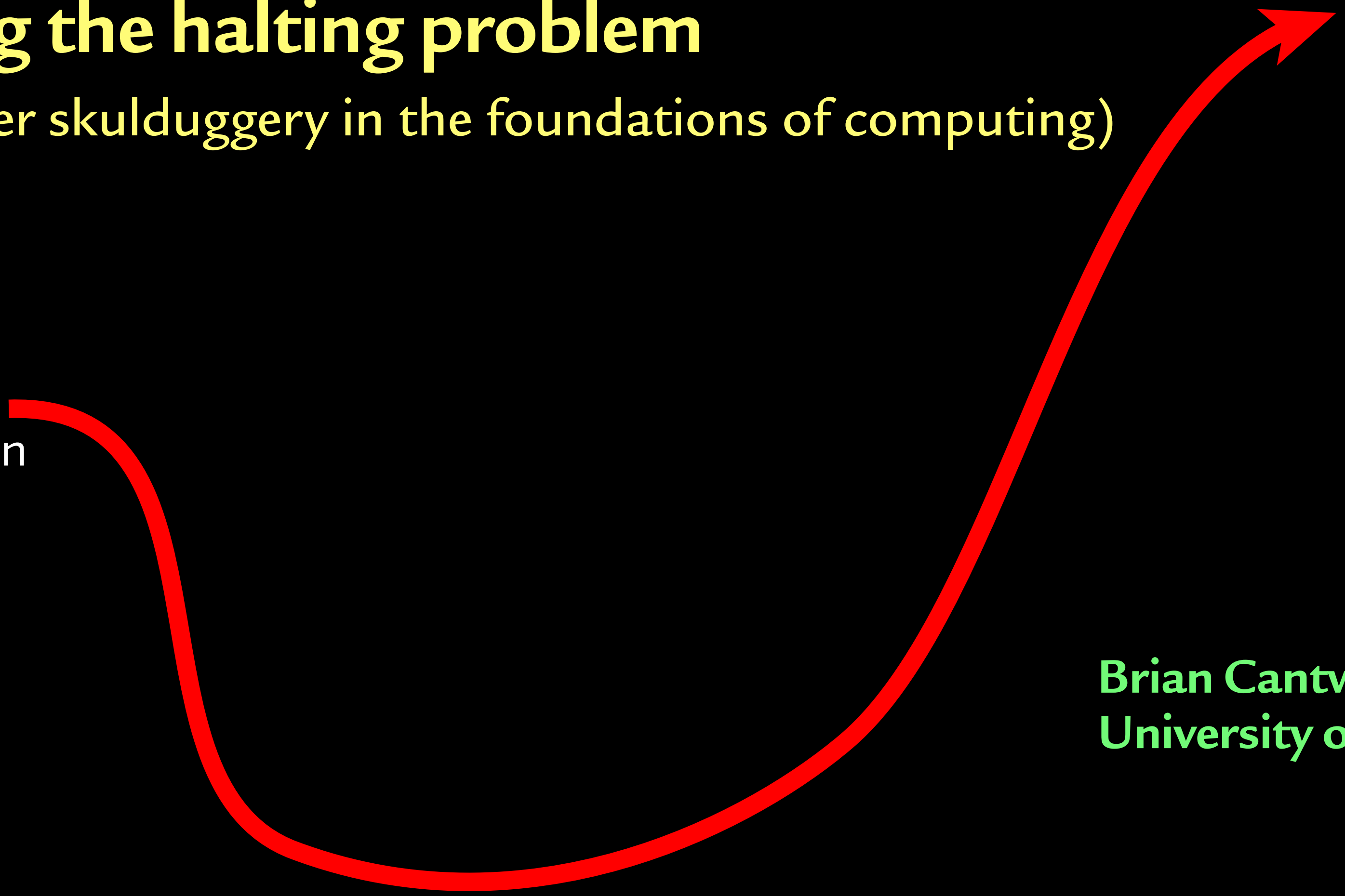
Solving the halting problem

(and other skulduggery in the foundations of computing)



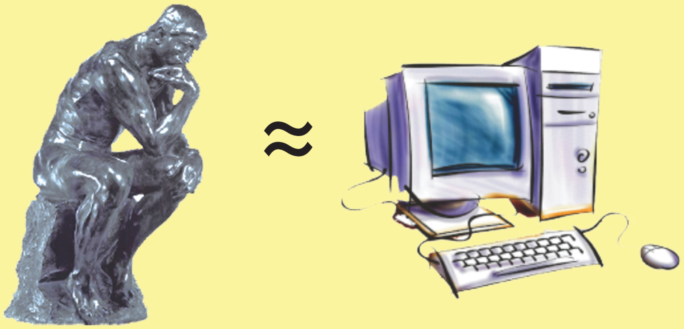
sensible
motivation

mind-numbing detail

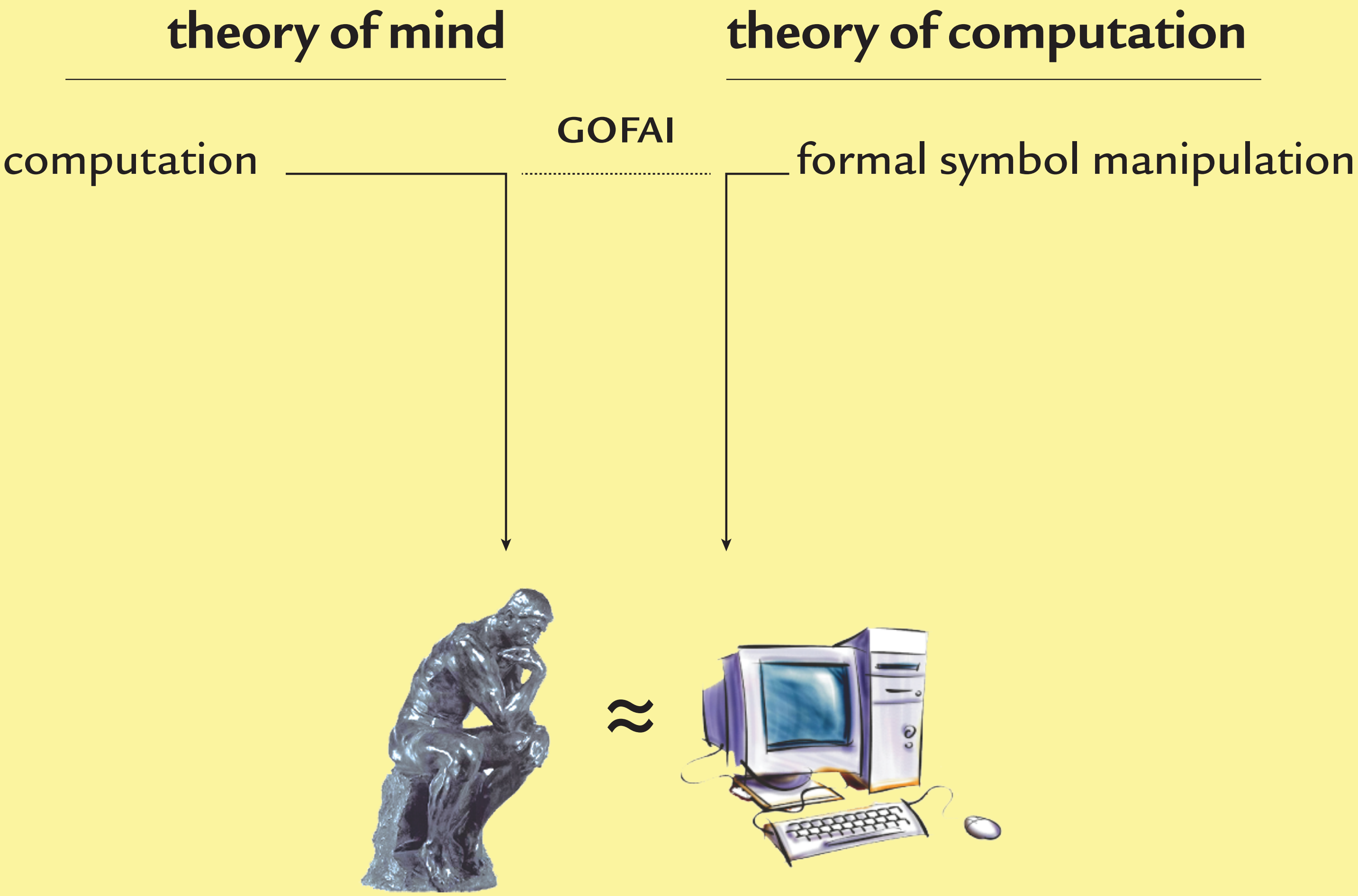
grand
conclusions!

Brian Cantwell Smith
University of Toronto

In 1967, as a university student, I was intrigued by the idea that people might be computers.



People thought computers were formal symbol manipulations, and so people if were computers, they must be formal symbol manipulators too. The idea was called "GOFAI"—Haugeland's name for "good old-fashioned artificial intelligence"



theory of mind

theory of computation

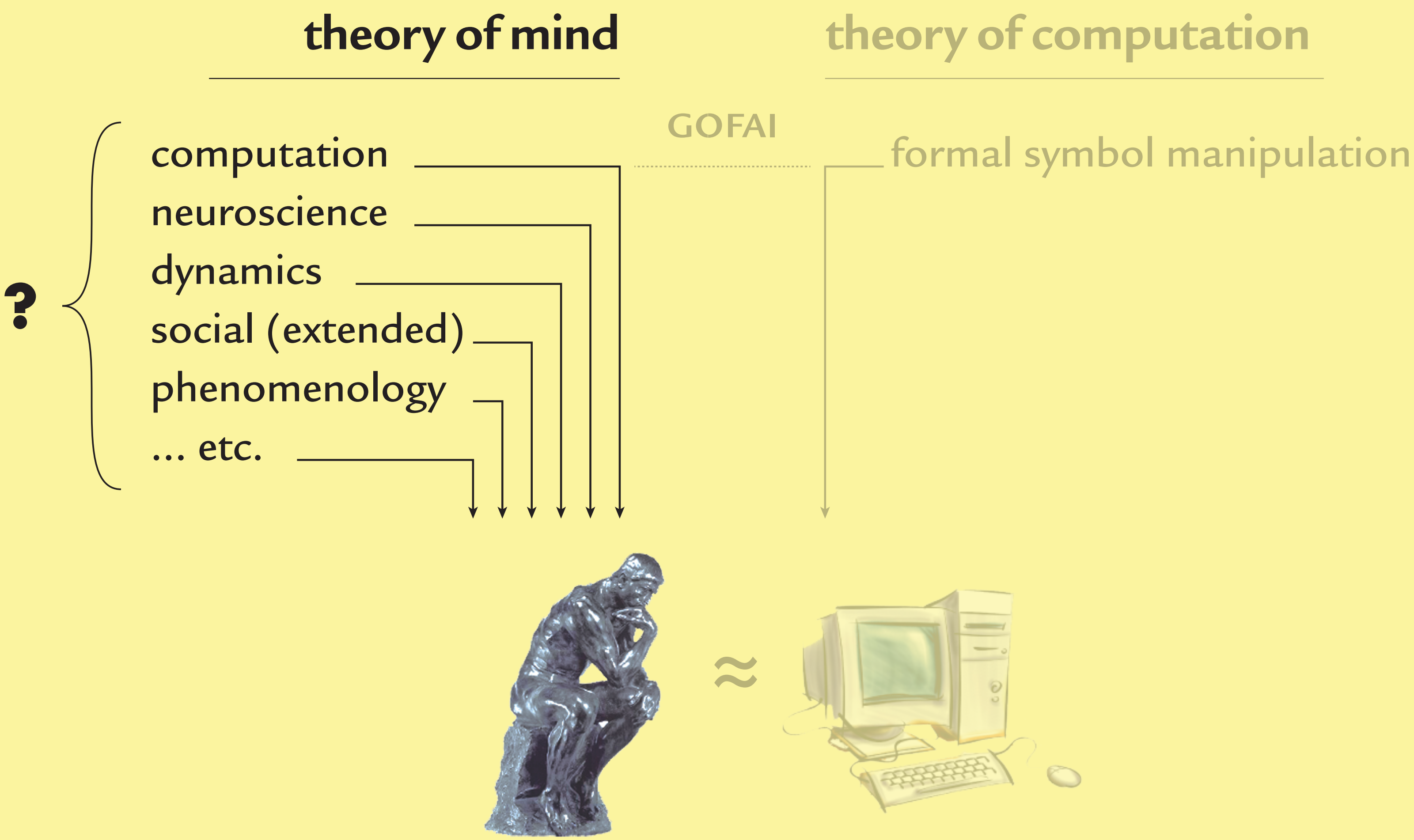
GOFAI

computation

formal symbol manipulation

≈

By the mid 1980s or so, GOFAI was deemed to have failed. Most cognitive scientists set computation aside, and turned to other theories of mind...the relative merits of which are still very much debated today.



theory of mind

theory of computation

?

computation

neuroscience

dynamics

social (extended)

phenomenology

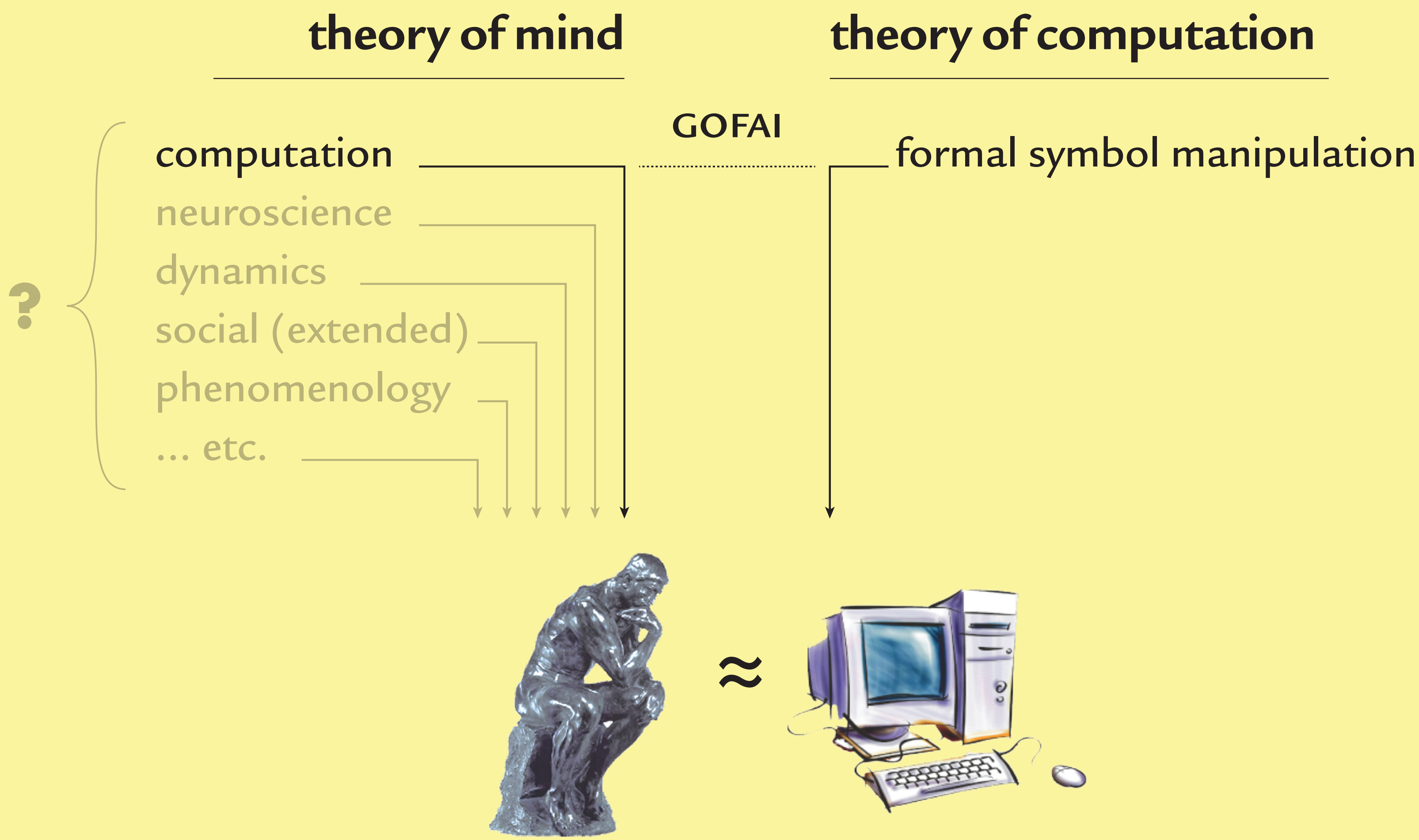
... etc.

GOFAI

formal symbol manipulation

≈

Being by nature slow, I remained interested in the "people might be computers" hypothesis.



theory of mind

theory of computation

GOFAI

computation

formal symbol manipulation

?

neuroscience

dynamics

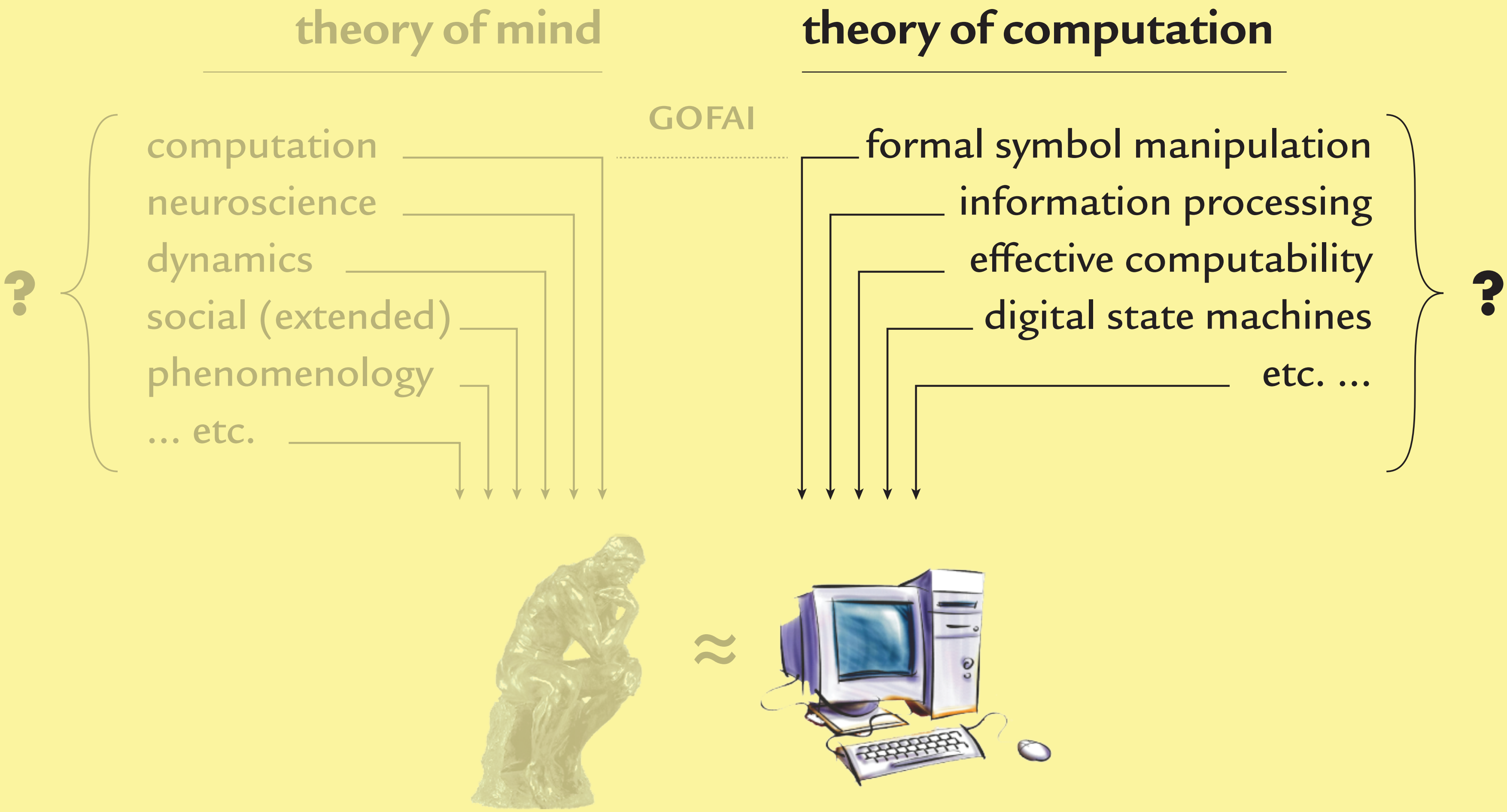
social (extended)

phenomenology

… etc.

≈

What preoccupied me was the assumption that computers were formal symbol manipulators. I wasn't sure that was true. *So I set temporarily set the cognitive case aside, and set out on an investigation of* what computers are.



theory of mind

theory of computation

GOFAI

?

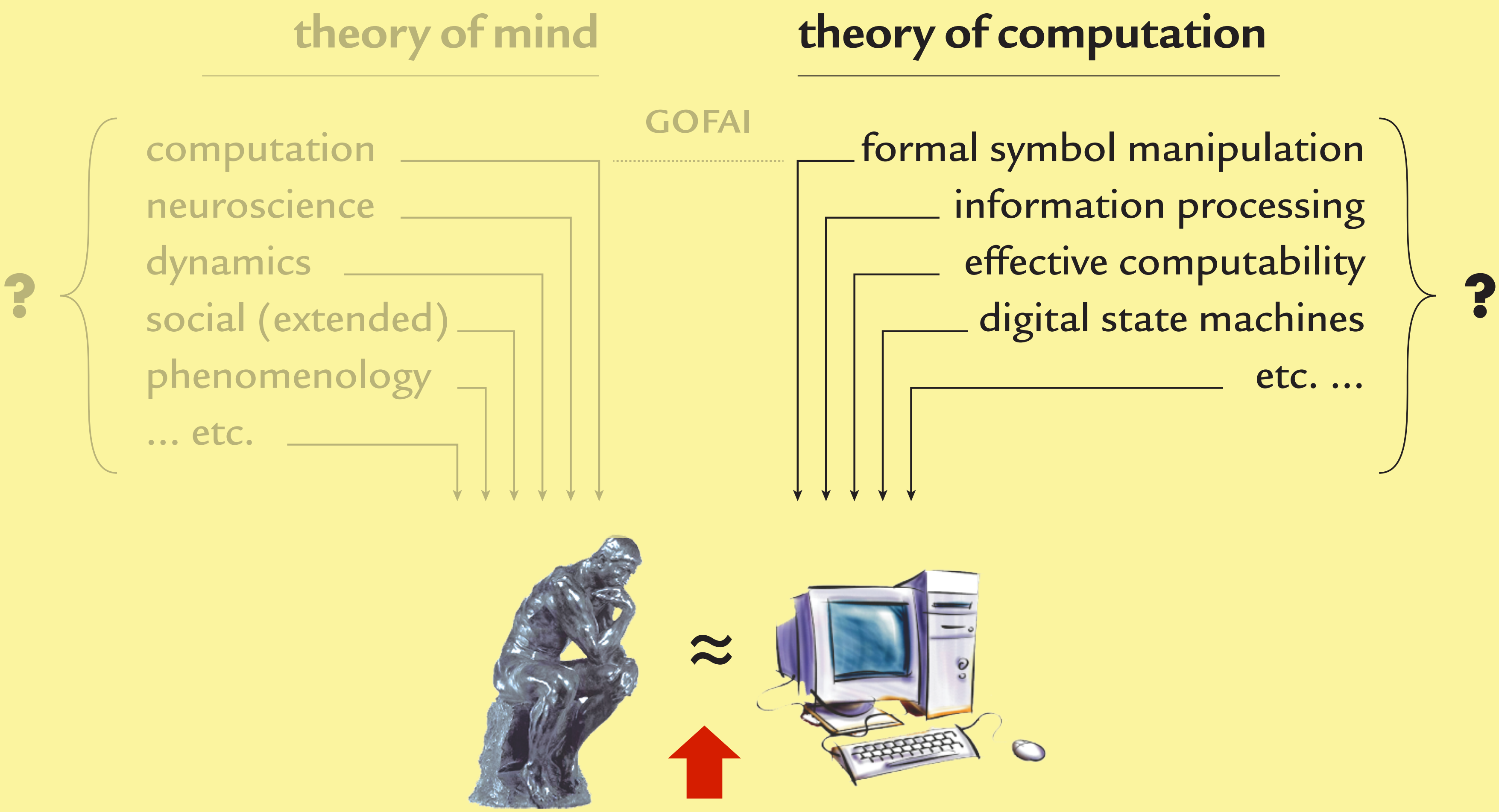
computation
neuroscience
dynamics
social (extended)
phenomenology
… etc.

formal symbol manipulation
information processing
effective computability
digital state machines
etc. …

?

≈

I've always taken the "people ≈ computers" hypothesis to be fundamentally empirical*: that people might be computers,* whatever computers are*. Until we have an adequate account of computation, we can't really assess the hypothesis.*



theory of mind

theory of computation

GOFAI

?

computation

neuroscience

dynamics

social (extended)

phenomenology

… etc.

formal symbol manipulation

information processing

effective computability

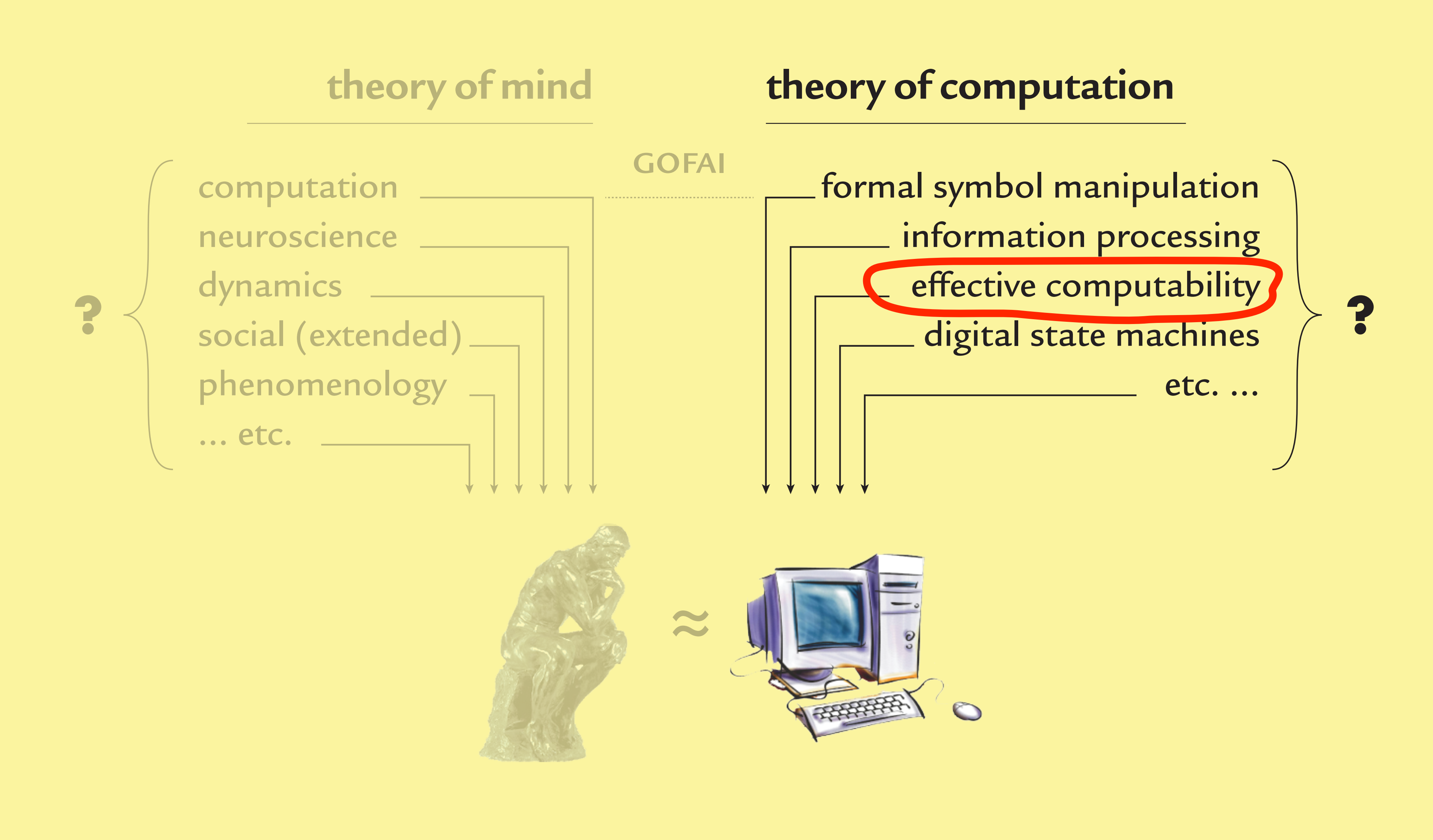
digital state machines

etc. …

?

≈

So I've spent the last 40 years trying to understand computation. Today I want to talk about one idea *about computers: the notion of "*effective computability*" ensconced in the official "theory of computation" or "theory of computability."*



theory of mind

theory of computation

?

computation

neuroscience

dynamics

social (extended)

phenomenology

… etc.

GOFAI

formal symbol manipulation

information processing

effective computability

digital state machines

etc. …

?

≈

Contents

1. Context
2. Turing machines
3. The halting problem (H)
4. Solving H
5. Morals for cognitive science

Contents

1. Context
2. ➡ Turing machines
3. The halting problem (H)
4. Solving H
5. Morals for cognitive science

known as the official
“theory of computation”

The theory of effective computability …

- A mathematical theory of computability constraints
- As important for showing what can’t be done
  (effectively, mechanically, by a “pure machine”)
  as for showing what can be done
- Most commonly formulated in terms of Turing Machines
- What sorts of constraints are they?
  - Mathematical?
  - Logical?
  - Physical?
  - Or is computation its own autonomous thing,
    subject to its own independent constraints?

During Creation…

On what day of the week did God make the computability constraints?

If

- Mathematics & logic were in place last week
- Laws of physics were created on Monday
- Semantics didn't arrive until Saturday

then

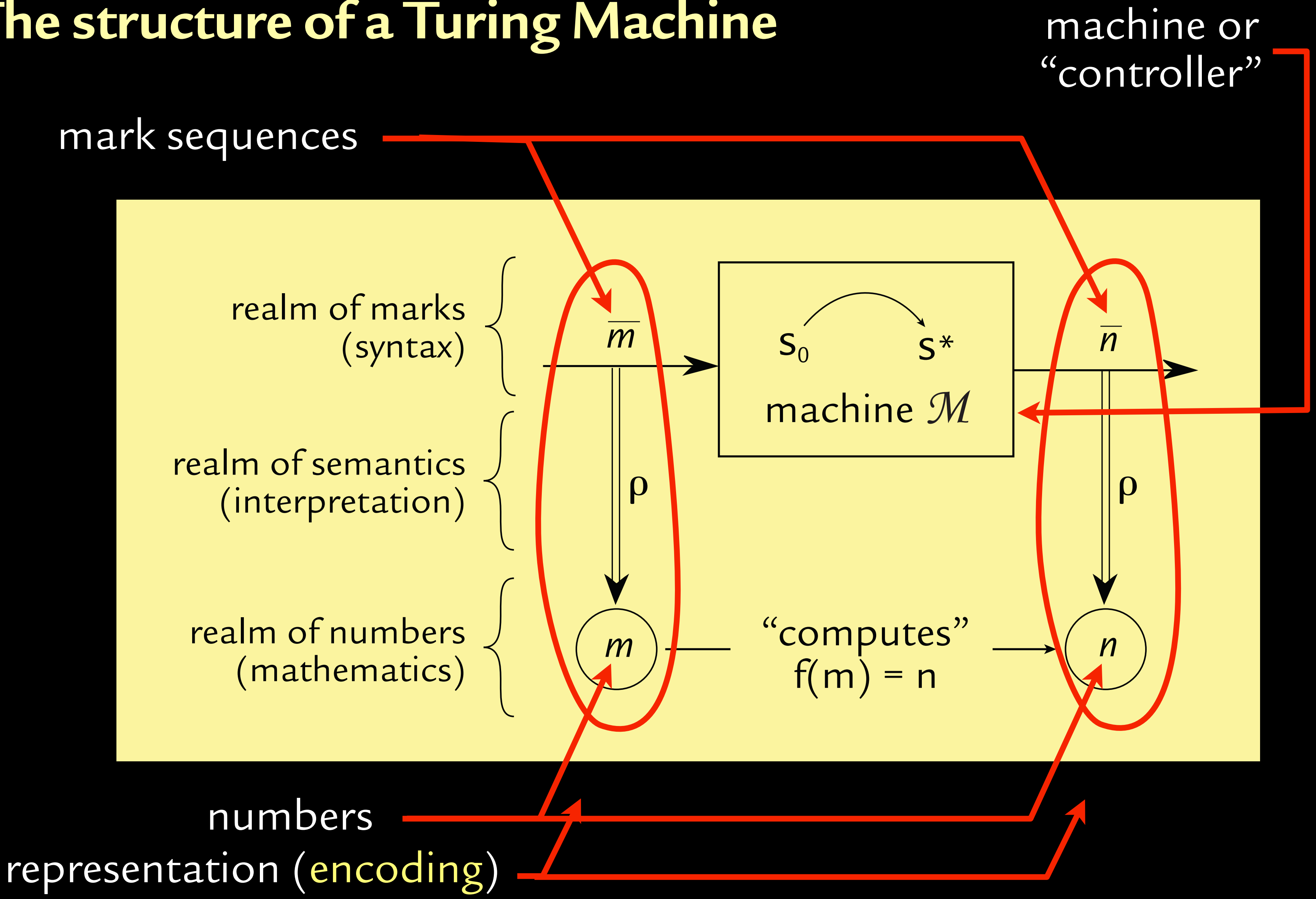
- On what day were the computability constraints established?

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
6:00 Let there be light!	9:00 Get to work on the firmament	ALL DAY: Make the earth	7:00 Create sun 21:00 Add moon	9:00 Birds & bees (also fish)	9:00 Animals & reptiles 12:00 Create man; get him to name the animals	ALL DAY: Rest!

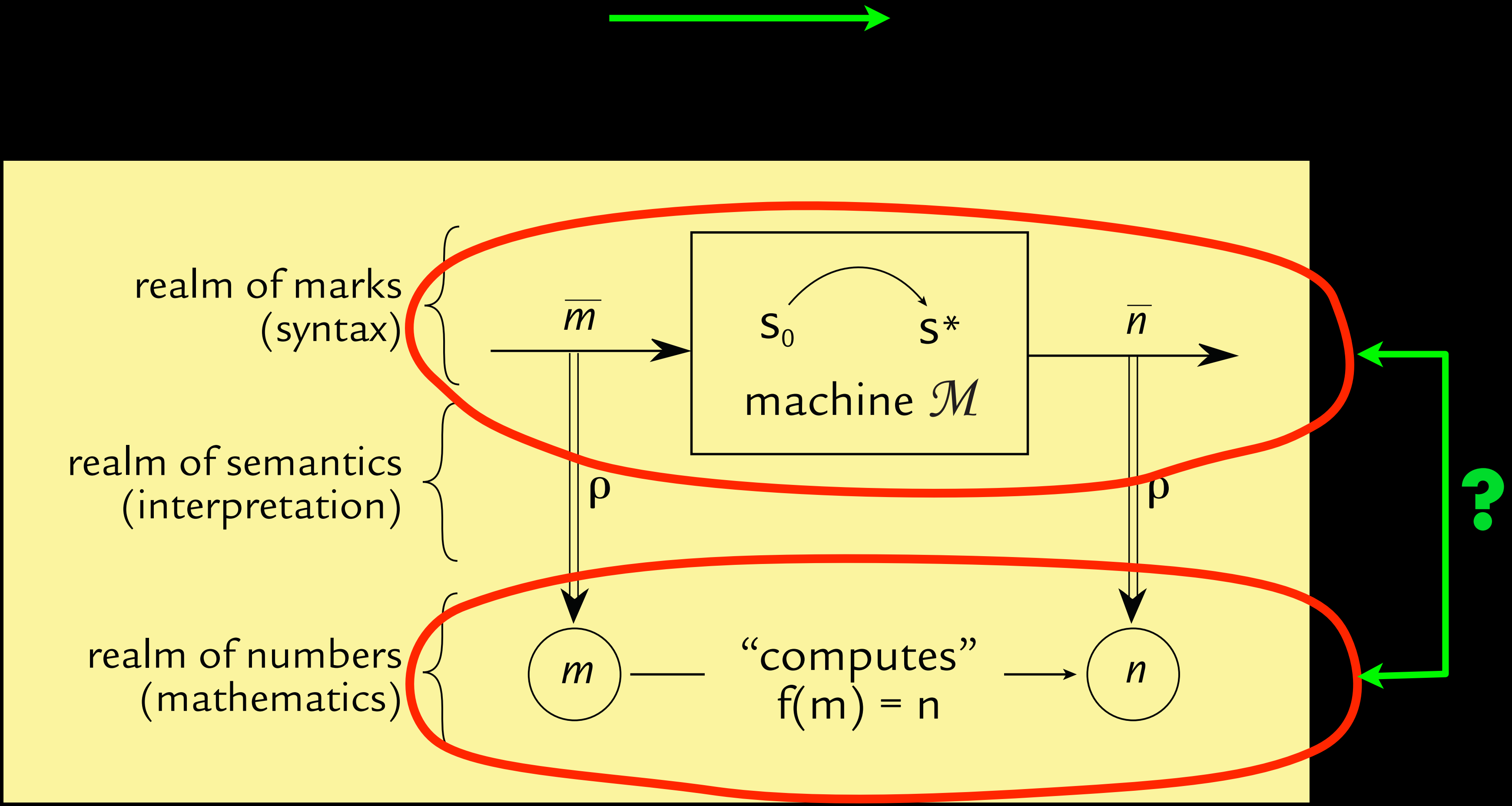
The theory of effective computability (cont'd)

- We can find out a lot about the nature of the constraints by analyzing the place they were introduced:
- Turing machines—and they limits on what they can do

The structure of a Turing Machine



Computation is unarguably horizontal… but at what level?



Terminology

Two uses of the word ‘compute’

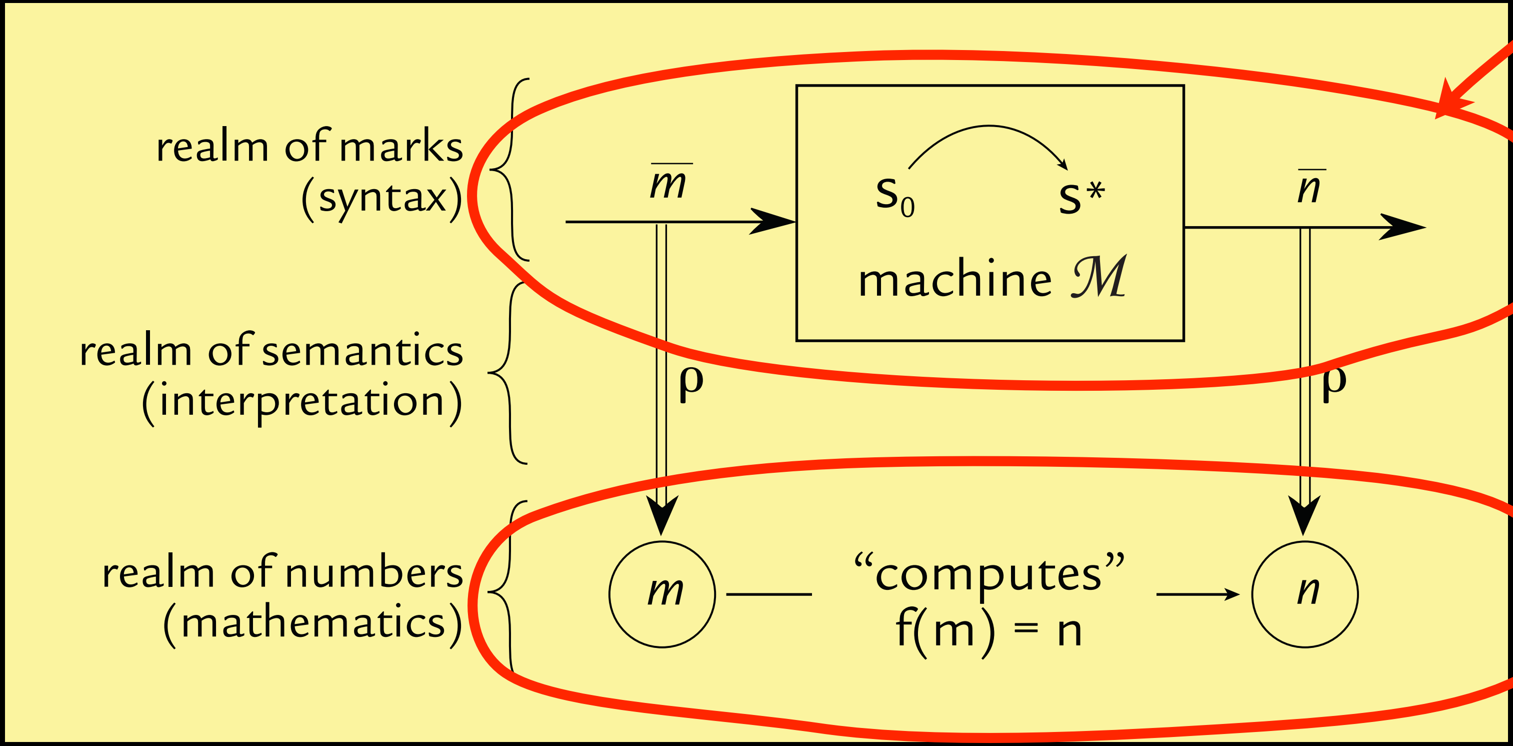
opaque (cf. ‘utter’): compute marks: ⟸ C.S.?
transparent (cf. ‘describe’): compute numbers: ⟸ logic?

$$\left(\begin{array}{rcc} & \text{sentence} & \text{car} \\ \text{utter} & ✔ & ✘ \\ \text{describe} & (✔) & ✔ \end{array}\right)$$

Amazingly, in the case of ‘compute’ there is no general agreement!

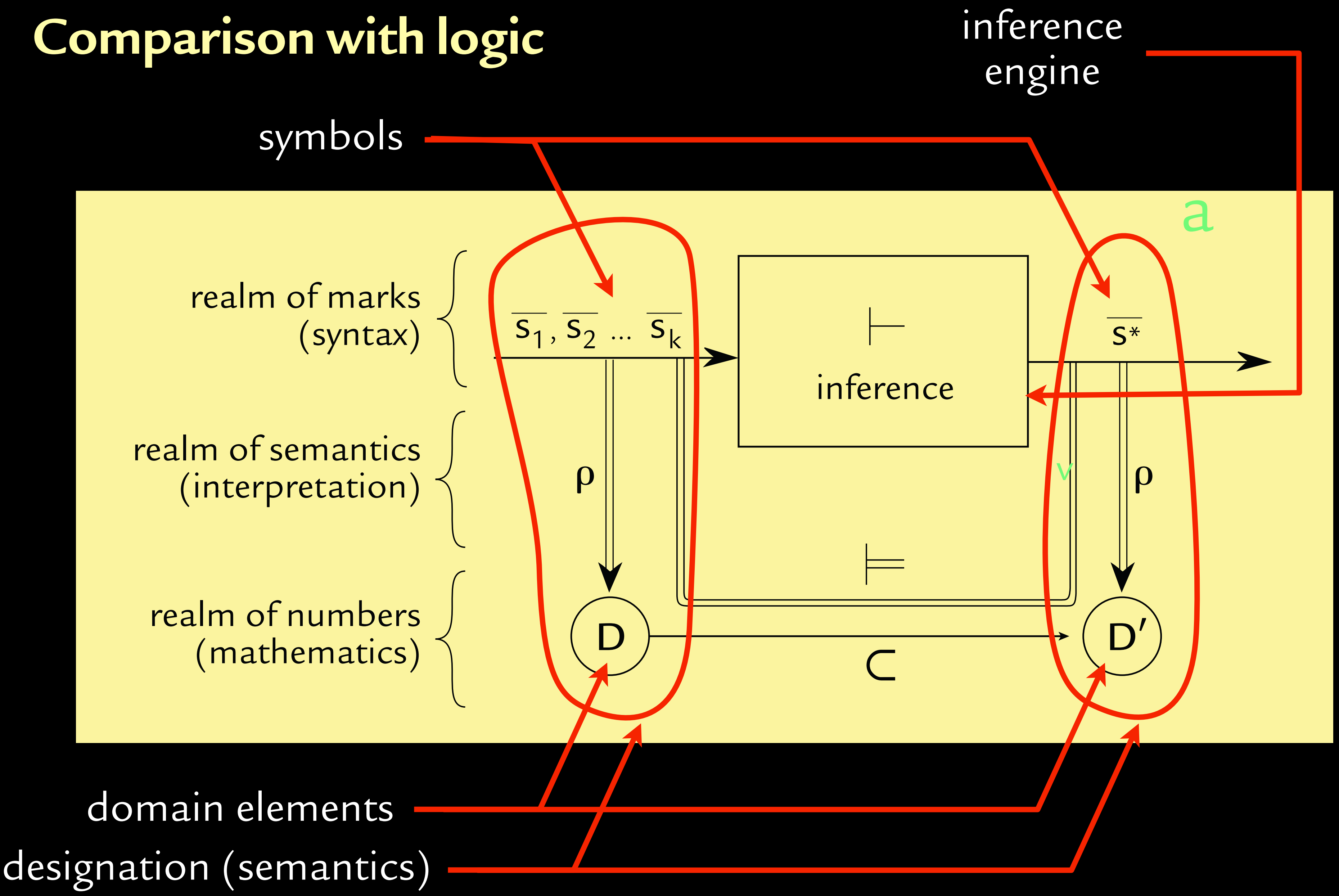
Terminology (cont'd)

$\boxed{\mathcal{E}_p}$ Opaque reading: effective computability is physical



$\boxed{\mathcal{E}_m}$ Transparent reading: effective computability is logical or mathematical

Comparison with logic



inference engine

symbols

realm of marks (syntax)

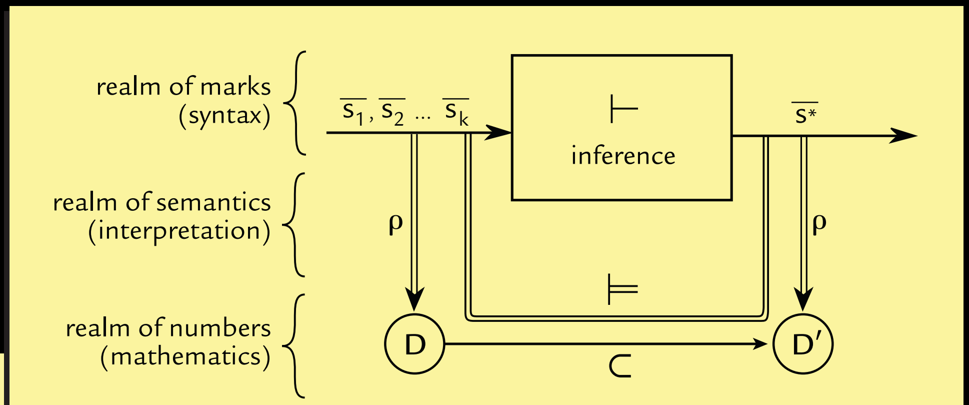
realm of semantics (interpretation)

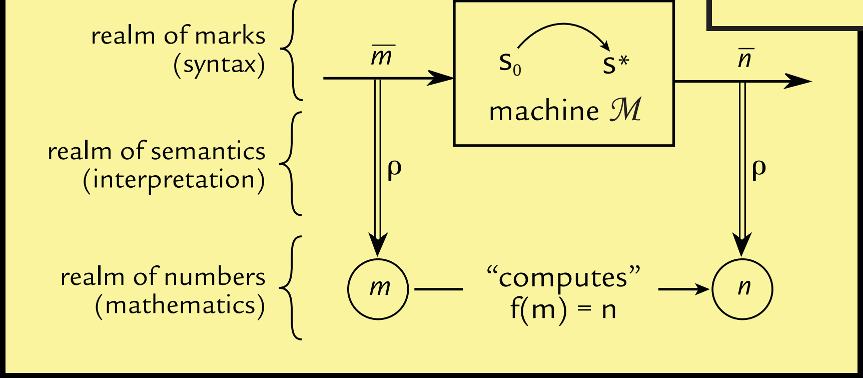
realm of numbers (mathematics)

domain elements

designation (semantics)

Similarities between Turing Machines & Logic

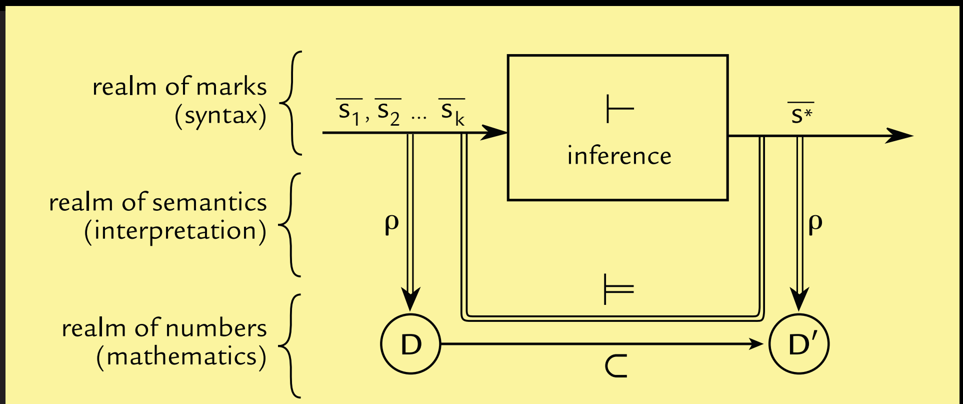


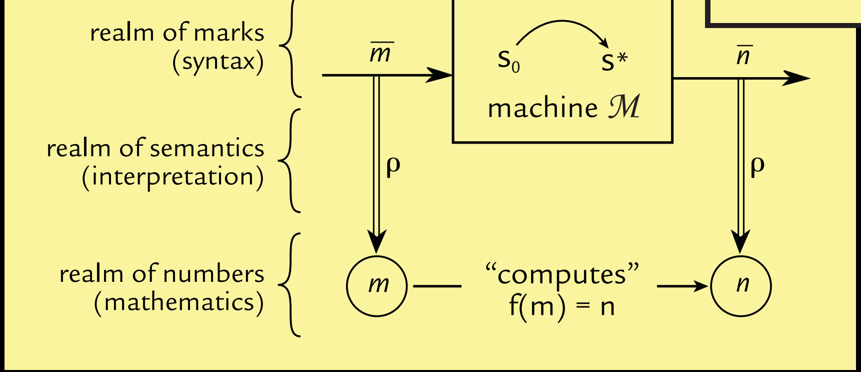


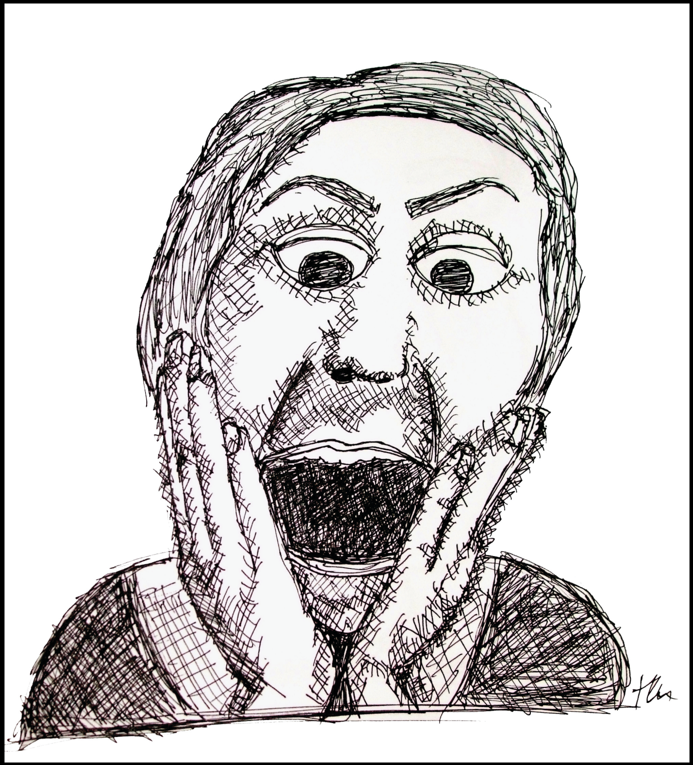
The fundamental truth about symbolic systems

1. Complex pattern of symbols with associated semantics (symbols/marks designate entities in a semantic domain
2. Fundamental dialectic
   a. System works "formally" (independent of semantics)
   b. System understood—and normatively governed—in terms of the semantics (designated entities)

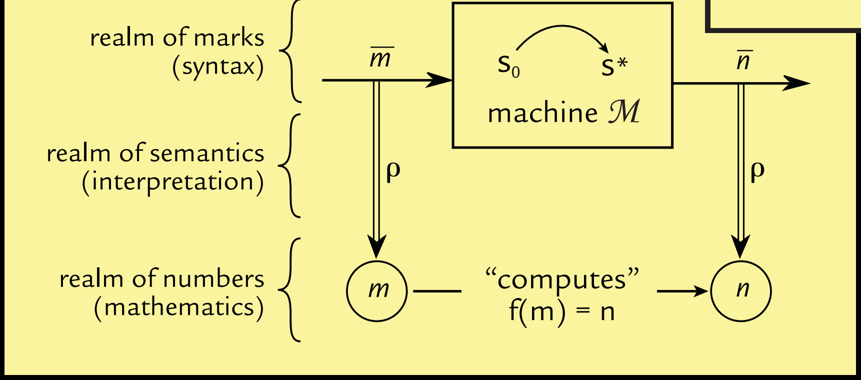
Similarities between Turing Machines & Logic *and minds!*







Similarities between Turing Machines & Logic *and minds!*



realm of marks (syntax)

realm of semantics (interpretation)

realm of numbers (mathematics)

$\overline{m}$

s_0 $s*$

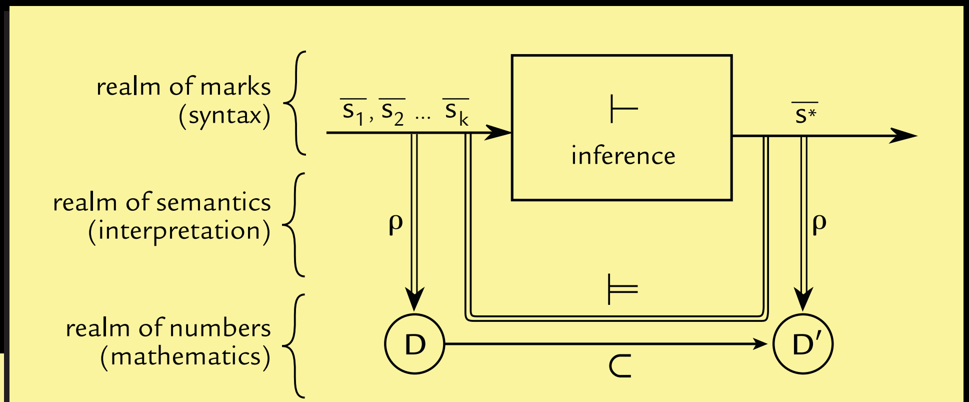
machine $\mathcal{M}$

$\overline{n}$

ρ

ρ

m — "computes" f(m) = n → n



realm of marks (syntax)

realm of semantics (interpretation)

realm of numbers (mathematics)

$\overline{s_1}, \overline{s_2} \ldots \overline{s_k}$

$\vdash$

inference

$\overline{s*}$

ρ

ρ

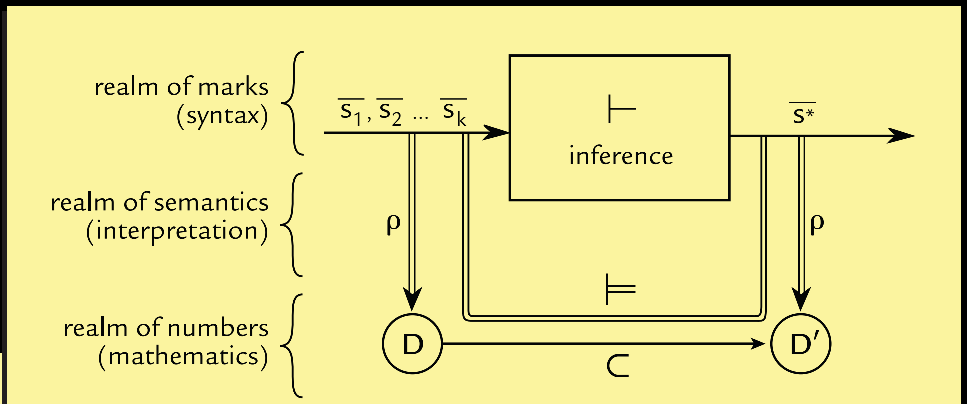
$\models$

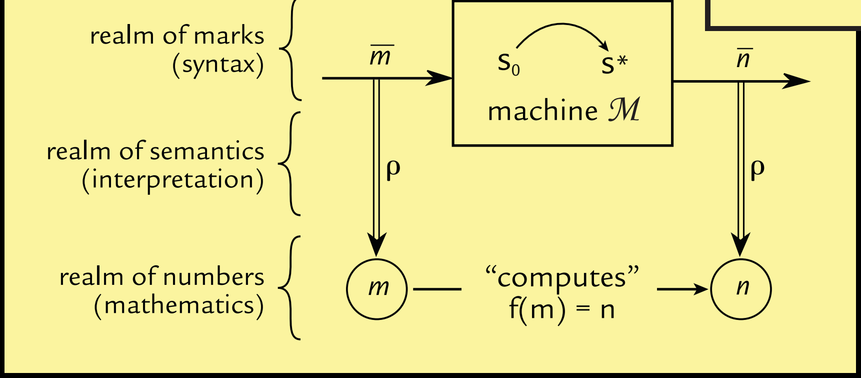
D

D′

$\subset$

Differences between Turing Machines & Logic



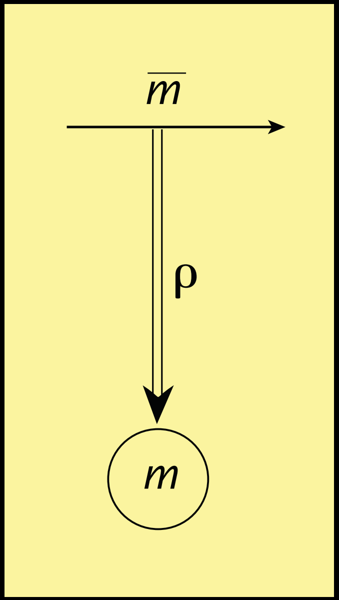


These (normative) constraints on the relations between syntax & semantics are the stuff & substance of logic

1. Logic: two domains (syntax & semantics), related by theorems:
   a. Soundness: semantics tracks syntax ⟸ "wanting what you get"
   b. Completeness: syntax tracks semantics ⟸ "getting what you want"
2. Computing: a single domain

Why the difference?

Reasonable Encodings



1. Why the difference?
2. The answer is buried in “reasonable encodings”
   a. The way semantic elements (“D”) are represented on the tape (ρ)
   b. Everyone uses reasonable encodings; almost no one talks about them
3. Universal agreement
   a. If you don’t use reasonable encodings, you get strange results
   b. E.g.: represent numbers as (numerals designating) their prime factors

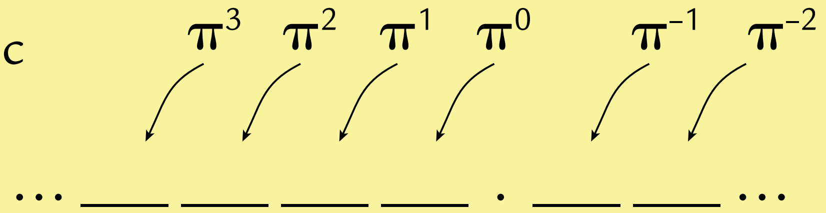
Prime factor representation:

1,745,308,631,353 = <7, 7, 17, 47, 563, 79181>

Reasonable Encodings (cont'd)

c. E.g.: base π encodings

Base π arithmetic



π^3 π^2 π^1 π^0 π^{-1} π^{-2}

$\cdots$ ____ ____ ____ ____ . ____ ____ $\cdots$

Some arithmetic truths:

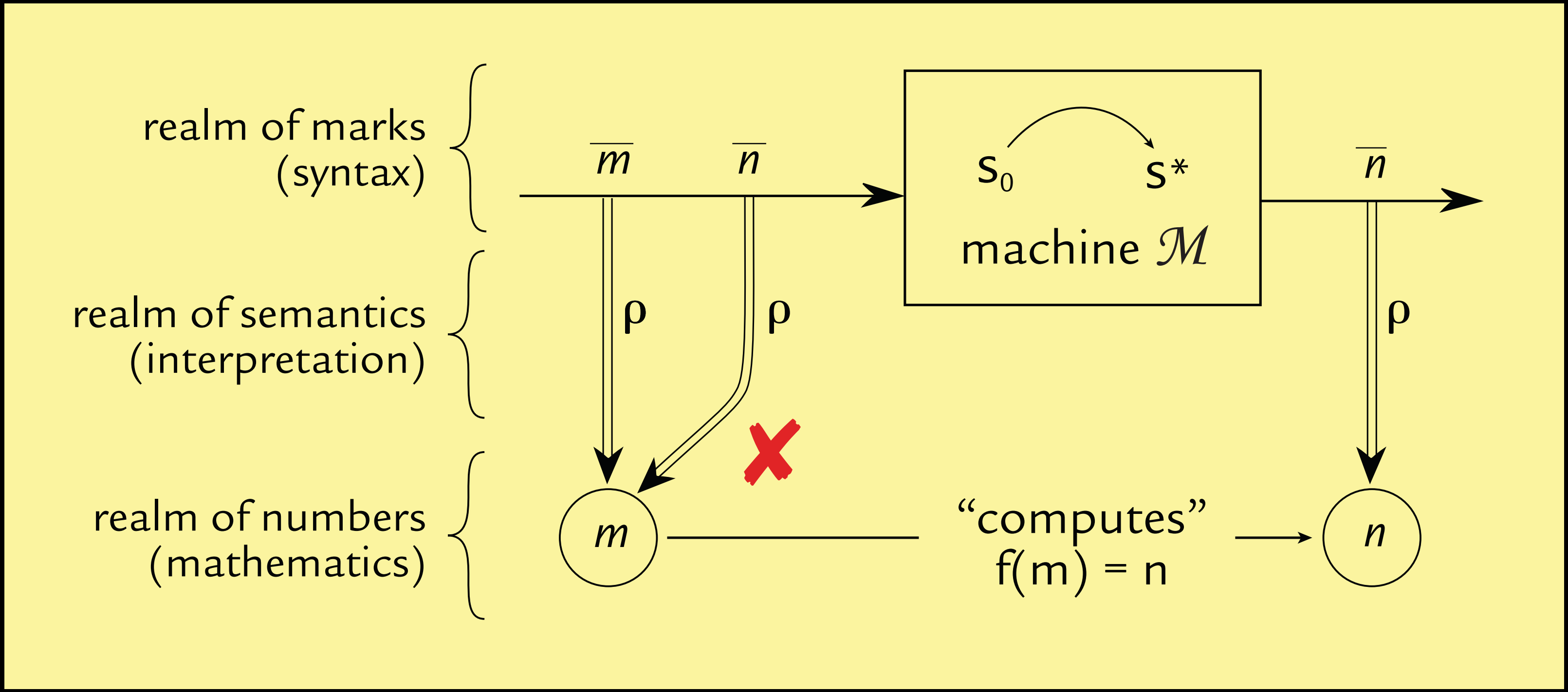
$$\begin{aligned} 1+1 &= 2 \\ 1+2 &= 3 \\ 10 &= 3.011021110020221130001\ldots \\ 3{+}1 &= 10.2020122021121110301\ldots \\ \pi &= 10 \\ \text{circumference} &= 10 \times \text{diameter} \end{aligned}$$

⟸ *i.e., shift the decimal one place to the left*

Reasonable Encodings (cont'd)

d. Can't even use simple equality (identity)

- And equality is not a very sophisticated semantic device!
- Cf. term arithmetic: 2 + 2 = 4



Constraints

1. In sum, in spite of the similarity of structure (mechanical operations on a domain of symbols designating entities in a semantic domain), when one looks hard logic & computing start to look very different

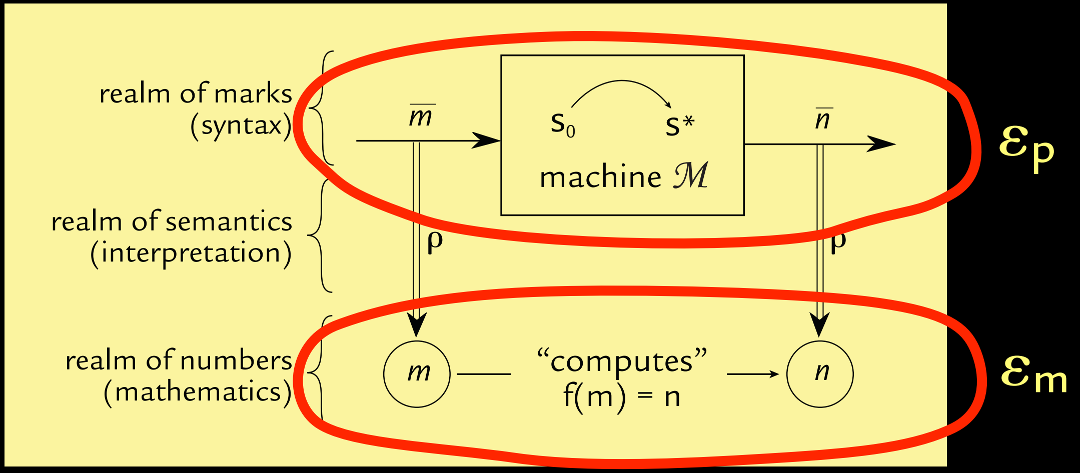
   a. In computing, representation ("reasonable encoding") is ridiculously restricted, compared to even very simple logics

   b. The stuff & substance of logic focuses on relations between the domains, which computing doesn't seem to be concerned about

2. So what is the "stuff & substance" of computing?

3. It has to do with effectiveness constraints on operations

4. That there are limits on both ε_p and ε_m is the most important result in all of computer science



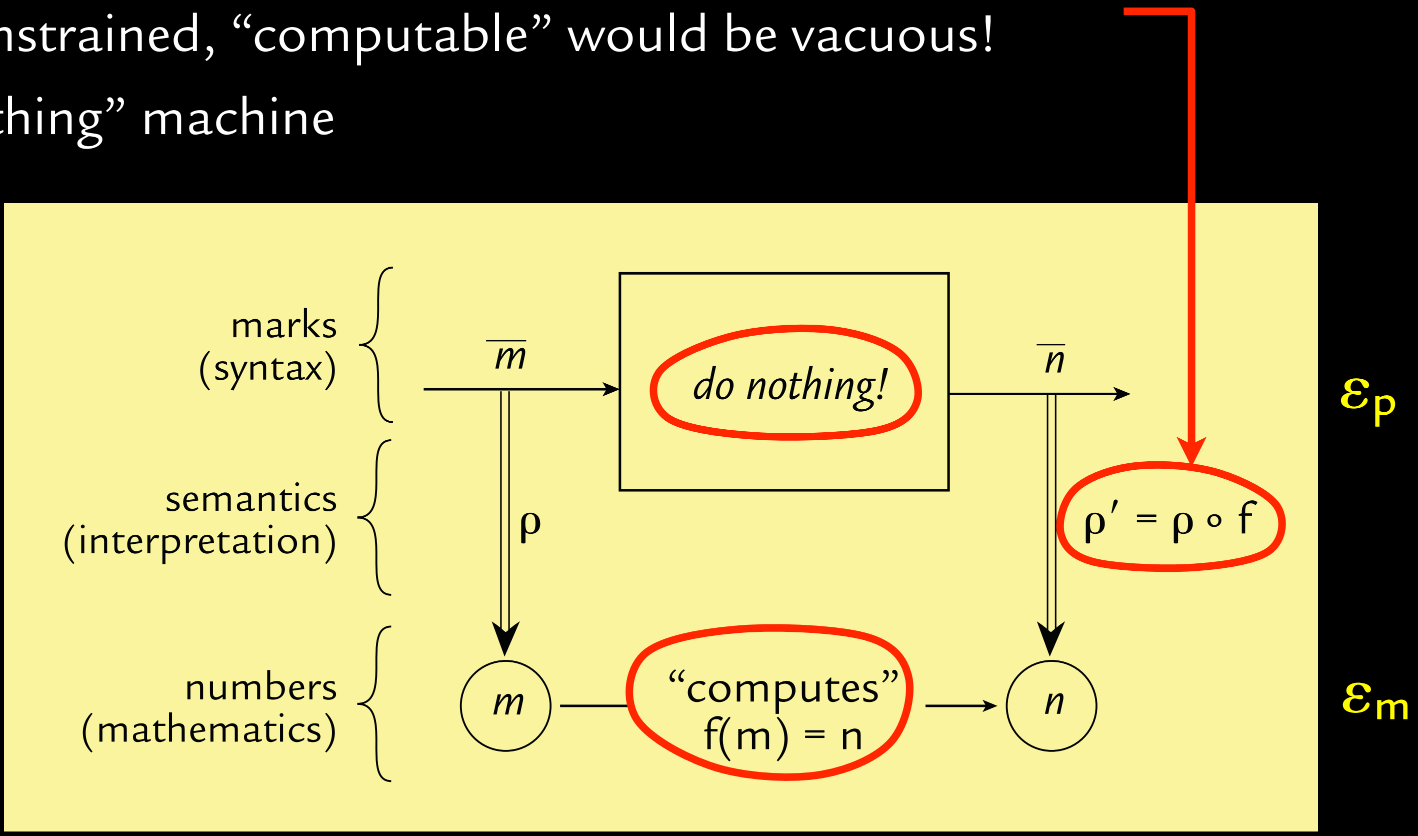
Constraints (cont'd)

5. These constraints must hold in several places
   a. Horizontally, on both "compute" operations ε_p & ε_m
   b. Vertically, on the representation relation ρ
6. If ρ weren't constrained, "computable" would be vacuous!
7. Cf. the "do nothing" machine

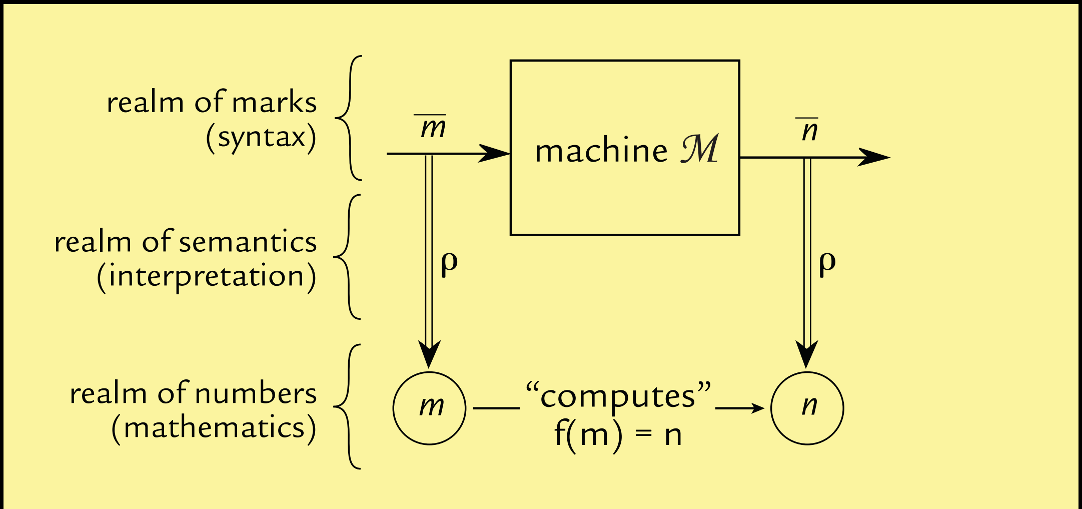
7 x 13?
7 x 13!

SEK10^9 question

what is the constraint on ρ?



Constraint on ρ



1. Can ρ (representation, encoding) be constrained in terms of effectiveness (ε_p & ε_m)?
2. No! … for a variety of reasons
   - — ε_p: wrong type (ε_p : mark ⇒ mark; ρ : mark ⇒ number)
   - — ε_m: circular (we need to define ε_m in terms of ρ!)
   - — ε_m: too broad! (allows the "do nothing" machine)
3. This leads to our first result:

R1: a) ρ must be *strictly weaker* than ε_m (i.e., $\rho \subset \varepsilon_m$)
b) Intuitively, the computable functions are $\sim[\varepsilon_p - \rho]$

Questions (that we need to answer)

(Vertical) questions: about encoding (ρ)

$Q \cdot \rho 1$ — What are the constraints on "reasonable" encoding?

$Q \cdot \rho 2$ — Why are they theoretically ignored?

$Q \cdot \rho 3$ — Should they be studied (put back on the table)?

(Horizontal) questions: about effectiveness (ε)

$Q \cdot \varepsilon 1$ — What is the fundamental character of the effectiveness ("computability") constraints?

$Q \cdot \varepsilon 2$ — Are these effectiveness constraints properties of

- The realm of marks (ε_p)? ⟸ Monday
- The realm of semantics (ε_s)? ⟸ Saturday
- The realm of mathematics (ε_m)? ⟸ Last week

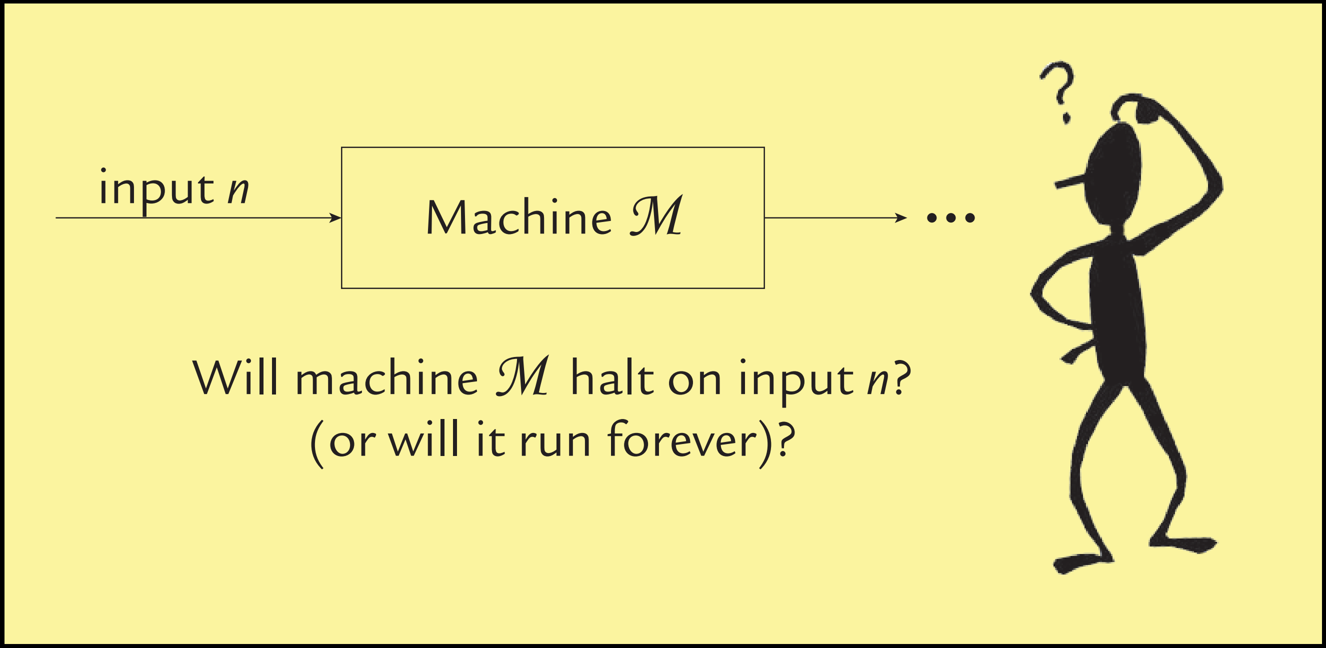
$Q \cdot \varepsilon 3$ — Can computability and complexity theory be conducted (as usual) at level of mathematics?

Contents

1. Context
2. Turing machines
3. The halting problem (H)
4. Solving H
5. Morals for cognitive science

The halting problem

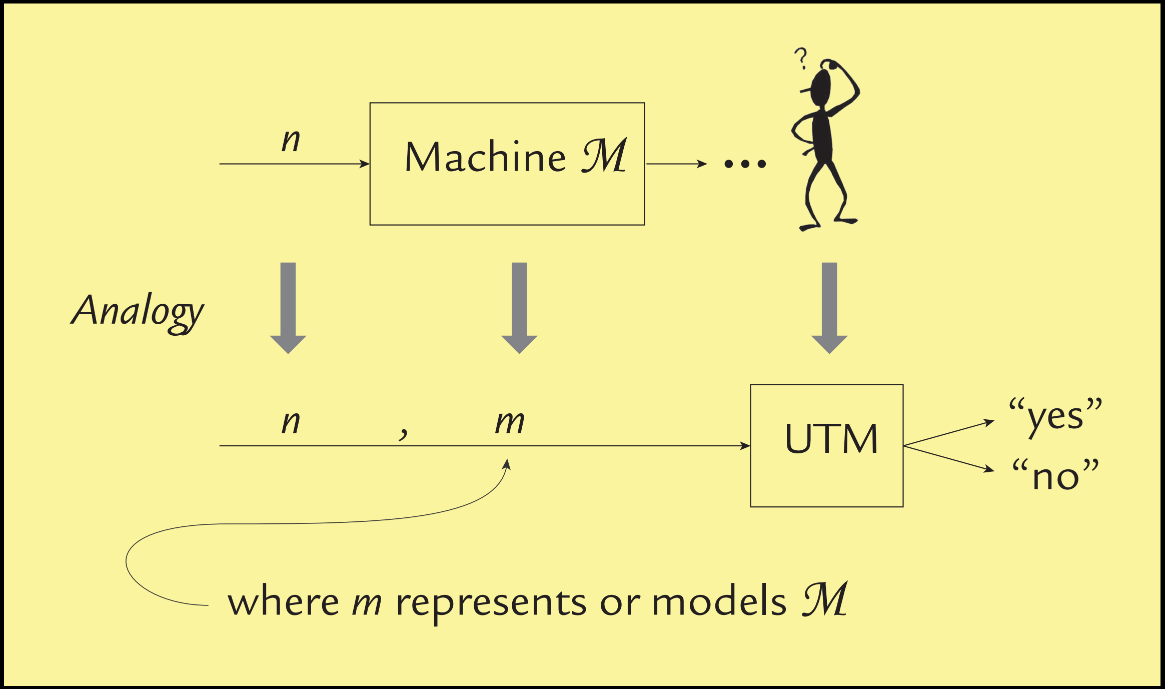
1. In general, given an input, machines (computers) will sometimes produce an output and then stop, but sometimes they will run forever, without ever "halting" and producing anything.
2. As Turing proved, it turns out to be impossible to decide whether an arbitrary machine will halt on an arbitrary input



Will machine $\mathcal{M}$ halt on input n?
(or will it run forever)?

Universal Turing machines

1. The proof consisted of two steps:
   a. He showed that a special "universal Turing machine" (UTM) could do "anything that could be done in a finite series of steps," by simulating any arbitrary other (Turing) machine
   b. He then proved that no Turing machine (and presumably no person, either!) could "decide" whether arbitrary machine M would halt given arbitrary input n.
2. Both steps can be challenged
3. Today I will focus on the second (a similar argument can be made against the first)

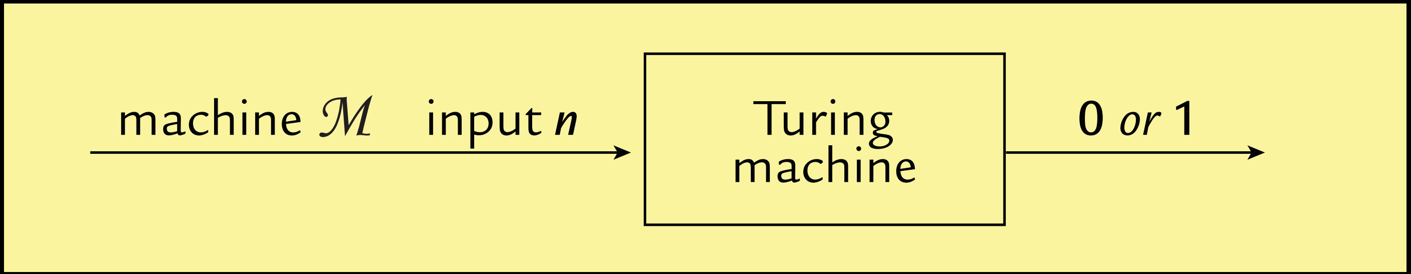


Solving the halting problem

1. Because we are testing limits, we need to proceed very carefully
2. Start with a simple characterisation of H:

> H1 — Given arbitrary Turing machine $\mathcal{M}$, and input n,
> decide whether or not $\mathcal{M}$ would halt on n

3. To "decide whether something is the case or not" is not well-defined, however—and "decide" is a ***semantical*** notion. (If there is a "decision," you can ask what the devision is *about*—and 'aboutness' is always a telltale sign of intentionality and semantics.)
4. To simplify, use 0 and 1, and try to build the following machine



machine $\mathcal{M}$ input n

Turing machine

0 *or* 1

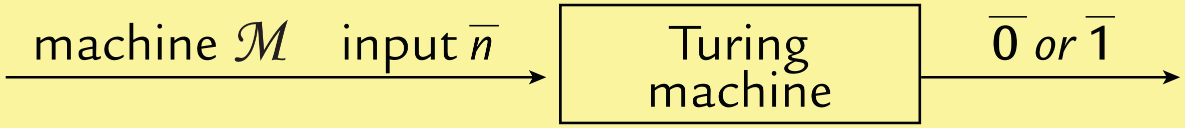
Solving the halting problem (cont'd)

5. That leads to the most common formulation of the problem:

   H2 — Given arbitrary Turing machine $\mathcal{M}$, and input n, produce 0 or 1, depending on whether or not $\mathcal{M}$ would halt on n

6. However machines can't take either numbers or machines as inputs; nor can they produce numbers as outputs

7. That's easy enough: use $\overline{0}$ to mean a symbol denoting 0 (i.e., $\rho(\overline{0})=0$), $\overline{1}$ to mean a symbol denoting 1, etc.:



machine $\mathcal{M}$ input $\overline{n}$

Turing machine

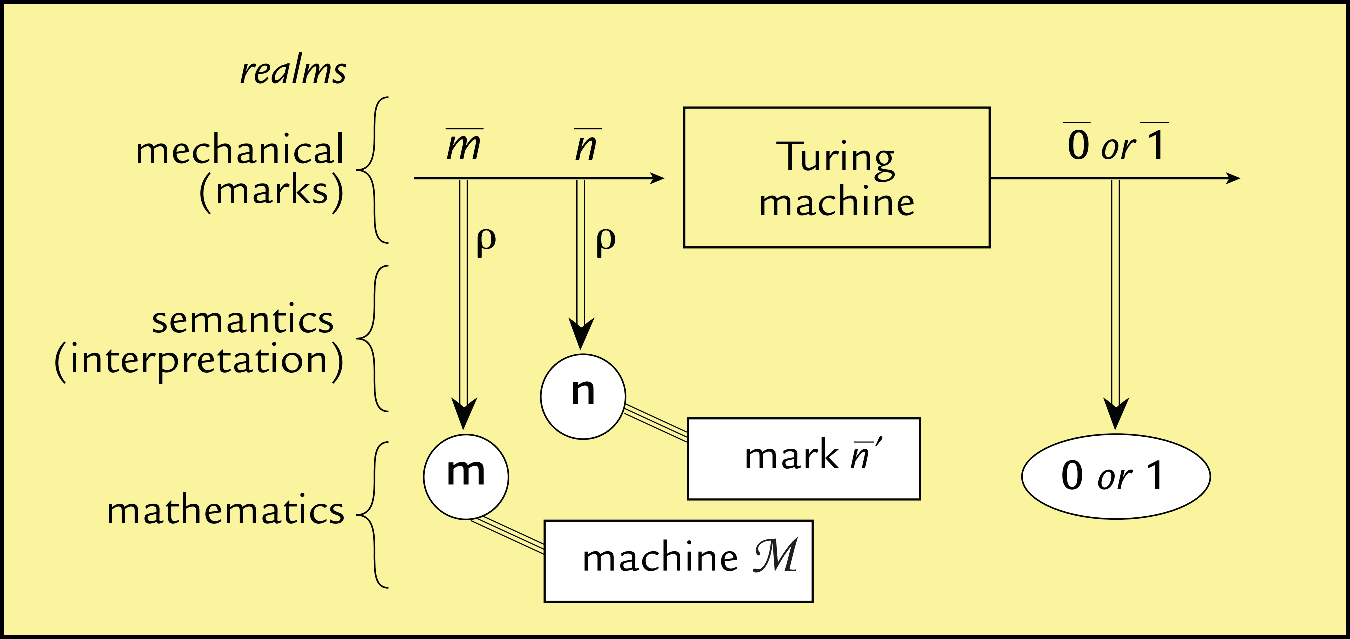
$\overline{0}$ *or* $\overline{1}$

8. This leads to the following problem statement:

   H3 — Given arbitrary Turing machine $\mathcal{M}$, and input $\overline{n}$, produce $\overline{0}$ or $\overline{1}$, depending on whether or not $\mathcal{M}$ would halt on $\overline{n}$

Solving the halting problem (cont'd)

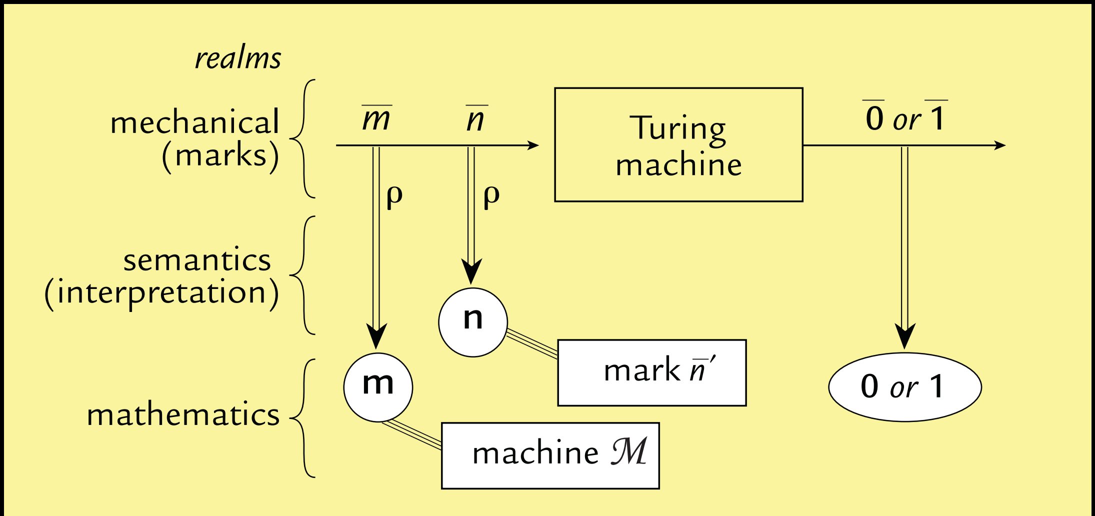
9. What about machine $\mathcal{M}$? *Machines* can't be inputs, either.
10. The strategy it to *model* $\mathcal{M}$ with a number m—typically a number that in turn "codes up" or models the set of quadruples* that Turing used to model Turing machines
11. We still can't input m, of course; we have to represent it with a symbol $\overline{m}$ representing m
12. A picture of what it comes to:



*A set of states, a set of symbol (types), an initial state, and a (finite) state transition function

Solving the halting problem (cont'd)

13. Try again, using a version that lays everything out explicitly:



H4 – Given as input marks $\overline{m}$ and $\overline{n}$, representing numbers *m* and *n*, respectively, produce as output marks $\overline{0}$ or $\overline{1}$, representing numbers 0 or 1, respectively, depending on whether the Turing machine $\mathcal{M}$ modelled by the set of quadruples μ coded by the number m would or would not halt if given as input the mark $\overline{\text{n}}'$ modelled by the number n′.

14. Note that **this makes** *3 different semantic relations explicit*

a.	Interpretation (ρ):	**marks (m, n)**	⇒ **numbers (m, n)**
b.	Modeling:	numbers (n, m)	⇒ quadruples (μ), marks (n')
c.	Coding:	quadruples (μ)	⇒ machines ($\mathcal{M}$)

Solving the halting problem (cont'd)

15. Because I will focus on output, we can simplify:
    a. Assume: $\mathcal{M}$ accepts marks in the same language (so $\overline{n} = \overline{n}'$)
    b. Assume: Relation between mark $\overline{m}$ and machine $\mathcal{M}$ unproblematic (call it 'indication')

16. Call the result H5:

> H5 – Given as input marks $\overline{m}$ and $\overline{n}$, representing numbers m and n, respectively, produce as output marks $\overline{0}$ or $\overline{1}$, representing numbers 0 or 1, respectively, depending on whether the Turing machine $\mathcal{M}$ (indicated by $\overline{m}$) would or would not halt if given $\overline{n}$ as input.

17. We can use H5 (finally!)

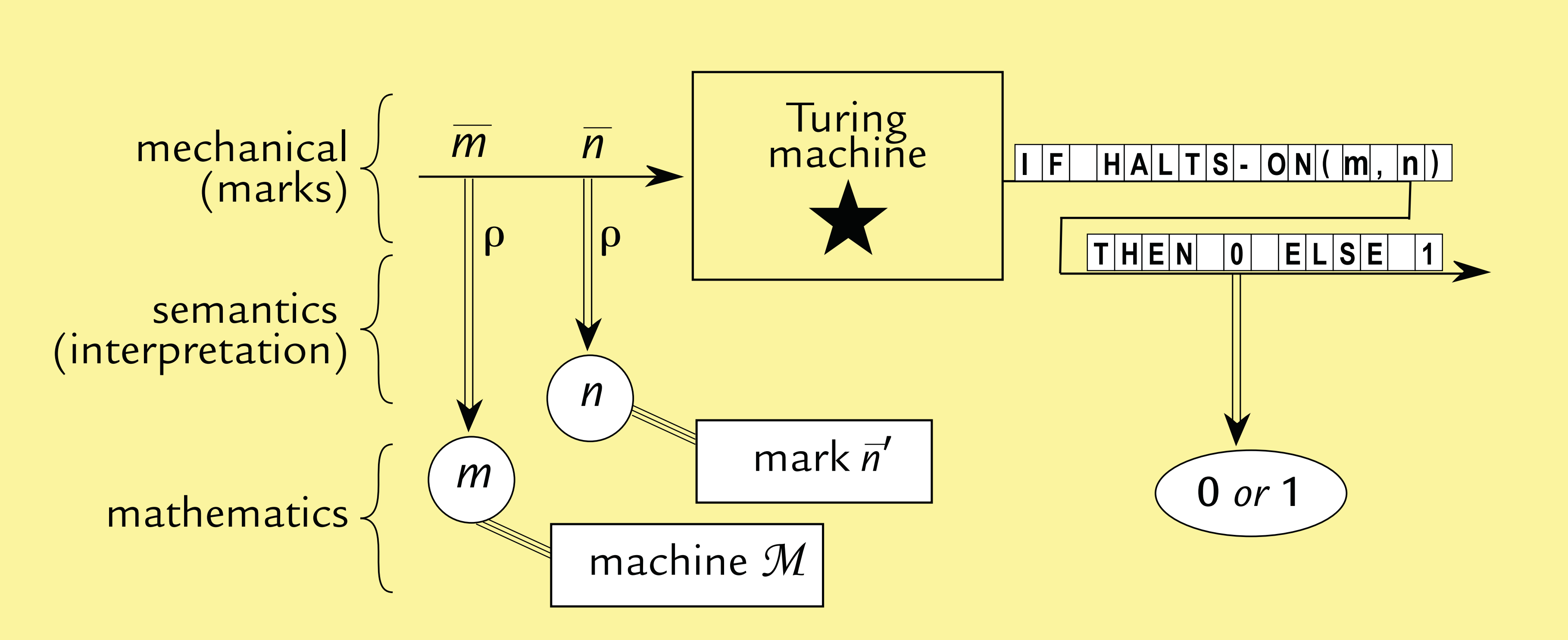
18. But there is a problem:

H5 is easy to meet!

Contents

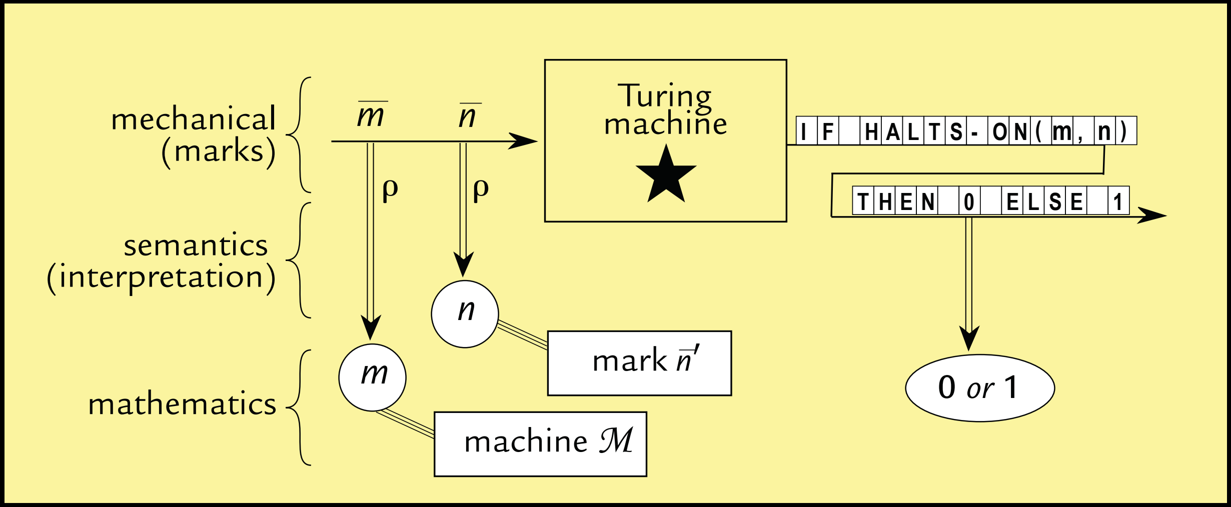
1. Context
2. Turing machines
3. The halting problem (H)
4. Solving H
5. Morals for cognitive science

Machine ★



★ *solves H5*

Machine ★



1. I.e., : ★ simply writes on the output tape:

   'IF HALTS-ON(__, __) THEN 0 ELSE 1'

2. and then substitutes in $\overline{m}$ and $\overline{n}$ in the blanks.

3. ★ is a cheat. *Of course!* But the question is why.

4. Many people will think the answer is *obvious*.

5. What I want to convince you of is:

   That obviousness is not as obvious as it looks!

6. We will learn a great deal by stepping through the issues slowly...

Catching ★

1. One thing that *is* obvious—and is also true!—is that there is a problem with the output. Call that output α.
2. Because the theory of computing is framed mathematically, let's start at the bottom level—at the level of functions, arguments, and values—and see whether we can catch ★ there.
3. Then, if as as that fails, we can move upwards...through semantics, and then finally to marks

Objection 1

1. Some people frame the following objection:

> O1 — Computer science has shown that the halts-on property "doesn't exist"—is not well-defined, can't be referred to, etc.

2. Objection O1 cannot be right

a. "Halts-on" must be well-defined, for the problem to make sense (if there weren't a metaphysical fact of the matter as to whether $\mathcal{M}$ halts on $\overline{n}$, the problem wouldn't exist as a problem)

b. There is no (semantical) problem in designating the "halts-on" property —since we do that in framing the problem (for example, it is referred to in every proof of the non-computability results).

c. So the mathematical facts are well-defined

Objection 2

1. Another common objection:

> O2 — To solve the problem, α must be a *numeral*, either 0 or 1

2. But α already *is* a numeral—satisfies the requirements on being a $\overline{0}$ or 1—since

a. Being a $\overline{0}$ just means that $\rho(\overline{0}) = 0$
Being a $\overline{1}$ just means that $\rho(\overline{1}) = 1$

b. α already satisfies this

  i. If $\mathcal{M}$ halts on n, then $\alpha_{m,n}$ designates 0 — i.e., $\rho(\alpha_{m,n}) = 0$

  ii. If $\mathcal{M}$ doesn't halt on n, $\alpha_{m,n}$ designates 1 — i.e., $\rho(\alpha_{m,n}) = 1$

Objection 3

1. It looks as if everything is OK at the bottom row (mathematics)
2. A third objection focus on the middle row: of semantics, saying:

O3 — The ρ that ★ uses is too complicated. Semantics (ρ) should be simple.

3. All α is, though, is a simple name, plus "if...then...else"—which is an extremely simple operator (far simpler than the ones people need in order to understand the halting problem)
4. Moreover, the "if...then...else" is unnecessary. We could change ★ so that it simply outputted HALTS-ON(__, __) THEN 0 ELSE, and substitute in the $\overline{m}$ and $\overline{n}$
5. So objection 3 also fails

Result 2

1. The failure of objection 3 shows an interesting power relation among human interpretation functions, effectively computable functions, and reasonable encodings.
2. We can call this our second result:

R2: It is intrinsic to the coherence of the halting problem, and to the notion of effective computability more generally, that if

a) ρ is the (class of) interpretation functions mapping mark sequences in “reasonable encodings” on tapes onto mathematical entities—functions, arguments, and values,

b) f_c is the class of effectively computable functions (ε_m), and

c) ρ' is the *human interpretation function*, from symbols, words and thoughts onto the world, including onto those same mathematical entities,

then

$$\rho \subset f_c \subset \rho'$$

Result 3

1. Moreover, because ★ satisfies H5, we know the answer to question $\mathrm{Q}\cdot\varepsilon 3$: whether the discourse of computability and complexity theory can be conducted *at the level of mathematics* (i.e., at the bottom level)

— That answer is *no!*

R3: The theory of effectively computable functions cannot be fully expressed at the (mathematical) level of functions, arguments, and values

Questions—and a first answer

(Vertical) questions: about encoding (ρ)

$Q\cdot\rho 1$ — What are the constraints on “reasonable” encoding?

$Q\cdot\rho 2$ — Why are they theoretically ignored?

$Q\cdot\rho 3$ — Should they be studied (put back on the table)?

(Horizontal) questions: about effectiveness (ε)

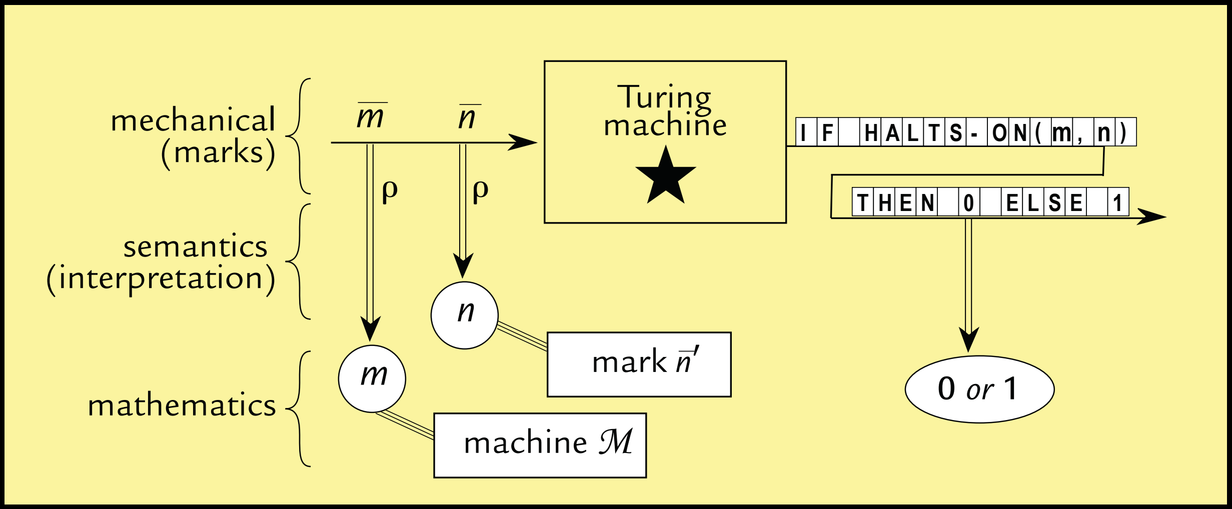
$Q\cdot\varepsilon 1$ — What is the fundamental character of the effectiveness (“computability”) constraints?

$Q\cdot\varepsilon 2$ — Are these effectiveness constraints properties of

- The realm of marks (ε_p)? $\Longleftarrow$ Monday
- The realm of semantics (ε_s)? $\Longleftarrow$ Saturday
- The realm of mathematics (ε_m)? $\Longleftarrow$ Last week

$Q\cdot\varepsilon 3$ — Can computability and complexity theory be conducted (as usual) at level of mathematics? ✗

Catching ★



Where are we?

1. We tried to catch ★ on the “bottom floor”: at the level of mathematics. That failed.

2. Then we tried to catch ★ on the “middle floor”: at the level of semantics. That failed too.

3. We need, finally, to do what was evident initially: go up to the “top floor,” and address the output α directly (i.e., $\alpha_{m,n}$ for each m, n)

Objection 4

The most obvious objection at the top row has two different sides:

O4a — ★ always produces the *same answer*, independent of whether $\mathcal{M}$ halts or not. It should produce *different answers in each case*.

O4b — ★'s outputs are *too* different. ★ should produce the *same answer* in each case where $\mathcal{M}$ halts on $\overline{m}$, $\overline{n}$; and also produce the same answer in each case where $\mathcal{M}$ does not halt on $\overline{m}$, $\overline{n}$—but a *different* "same answer" in the latter case than the former.

1. Objection O4a is certainly false. By design, *all the $\alpha_{\overline{m},\overline{n}}$ are different*
2. With respect to O4b, ★ can't literally output the *same token*, of course; what one must mean is that it must (in the appropriate cases) output *different tokens of the same answer type*. We need to spell this out more carefully …

Objection 4 (cont'd)

There are 3 criteria on what should be the same, and what should be different, between and among different answers $\alpha_{m,n}$. 2 are factual; 1 is counterfactual:

C1 — Different inputs should lead to same output, if they denote the same halting behaviour

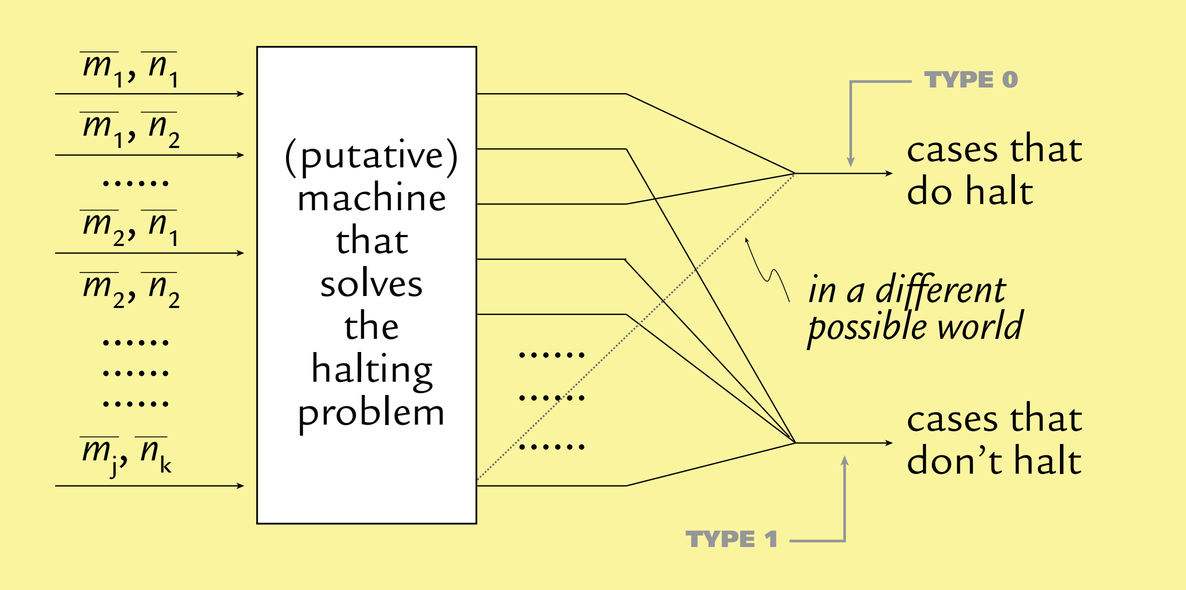
C2 — Different inputs should lead to different outputs, if they represent different halting behaviour; and

C3 — If m_i, n_k would have led to a different output—had the metaphysical, ontological, conceptual (whatever) facts about whether $\mathcal{M}_\mathrm{i}$ halts on $\mathrm{n_{ik}}$ been different—then in that case they should have led to a different output

In order to reach our standard level of pedantic fastidiousness, however, we have to restate this with explicit reference to the types involved.

Objection 4 (cont'd)

Try it again, making
the types explicit:



$\overline{m}_1, \overline{n}_1$

$\overline{m}_1, \overline{n}_2$

......

$\overline{m}_2, \overline{n}_1$

$\overline{m}_2, \overline{n}_2$

......

......

......

$\overline{m}_j, \overline{n}_k$

(putative) machine that solves the halting problem

......

......

......

TYPE 0

cases that do halt

in a different possible world

cases that don't halt

TYPE 1

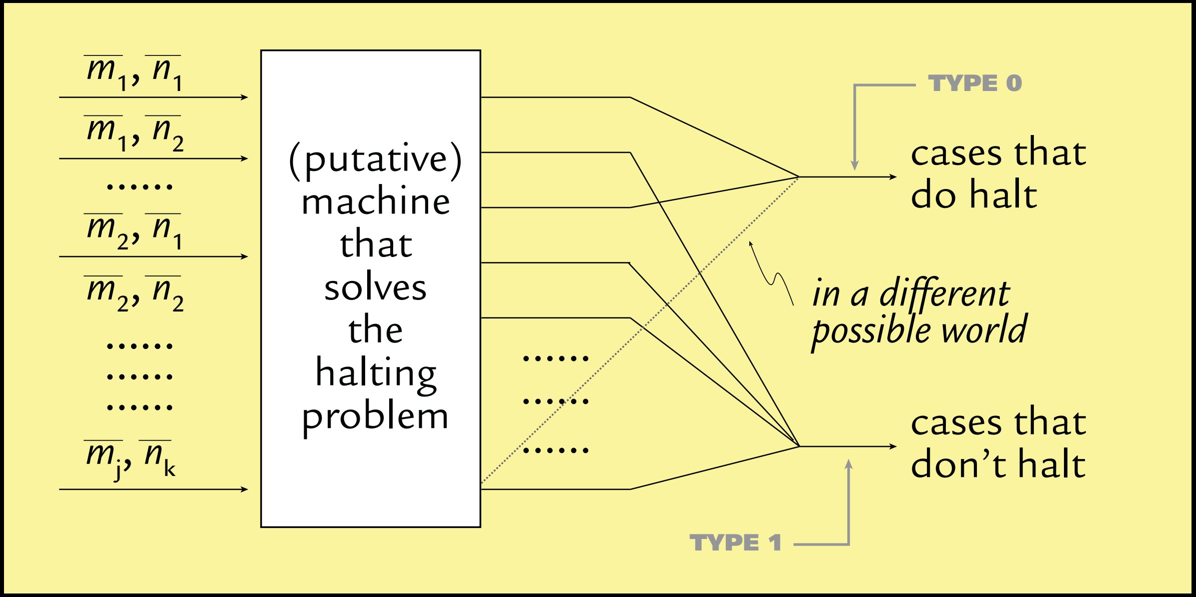
C1′ — Different inputs should lead to distinct tokens of the *same* output type, if they denote the *same* halting behaviour

C2′ — Different inputs should lead to distinct tokens of *different* output types, if they represent *different* halting behaviour; and

C3′ — If m_i, n_k would have led to a different output—had the metaphysical, ontological, conceptual (whatever) facts about whether $\mathcal{M}_i$ halts on n_{ik} been different—then in that case they should have led to tokens of different output types

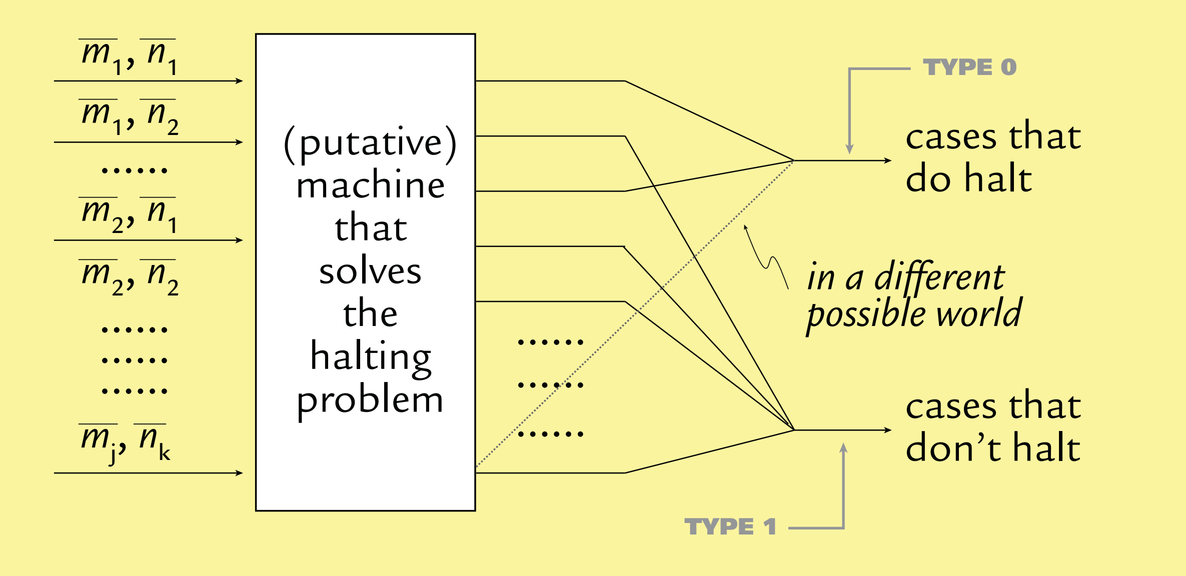
Objection 4 (cont’d)

Leads to yet another formulation of the problem:


$\overline{m}_1, \overline{n}_1$
$\overline{m}_1, \overline{n}_2$
……
$\overline{m}_2, \overline{n}_1$
$\overline{m}_2, \overline{n}_2$
……
……
……
$\overline{m}_j, \overline{n}_k$
(putative) machine that solves the halting problem
……
……
……
TYPE 0
cases that do halt
in a different possible world
cases that don’t halt
TYPE 1

H6 – Given as input marks $\overline{m}$ and $\overline{n}$, representing numbers m and n, respectively, produce as output tokens of mark types $\overline{0}$ or $\overline{1}$, representing numbers 0 or 1, respectively, depending on whether the Turing machine $\mathcal{M}$ (indicated by m) would or would not halt if given n as input, such that all tokens of $\overline{0}$ lead to a single output state or path, and all tokens of $\overline{1}$ lead to a *different* single output state or path.

Catching ★


$\overline{m}_1, \overline{n}_1$
$\overline{m}_1, \overline{n}_2$
……
$\overline{m}_2, \overline{n}_1$
$\overline{m}_2, \overline{n}_2$
……
……
……
$\overline{m}_j, \overline{n}_k$
(putative) machine that solves the halting problem
……
……
……
TYPE 0
cases that do halt
in a different possible world
cases that don't halt
TYPE 1

We aren't there yet!

1. Objection 4b still fails!
2. ★ already meets this condition H6, (i.e., even when elaborated in terms of criteria C1′—C3′.
3. The problem is with the terms 'same' and 'different'
4. Sameness & difference (identity) are always with respect to some metric of equivalence. Thus suppose we define:

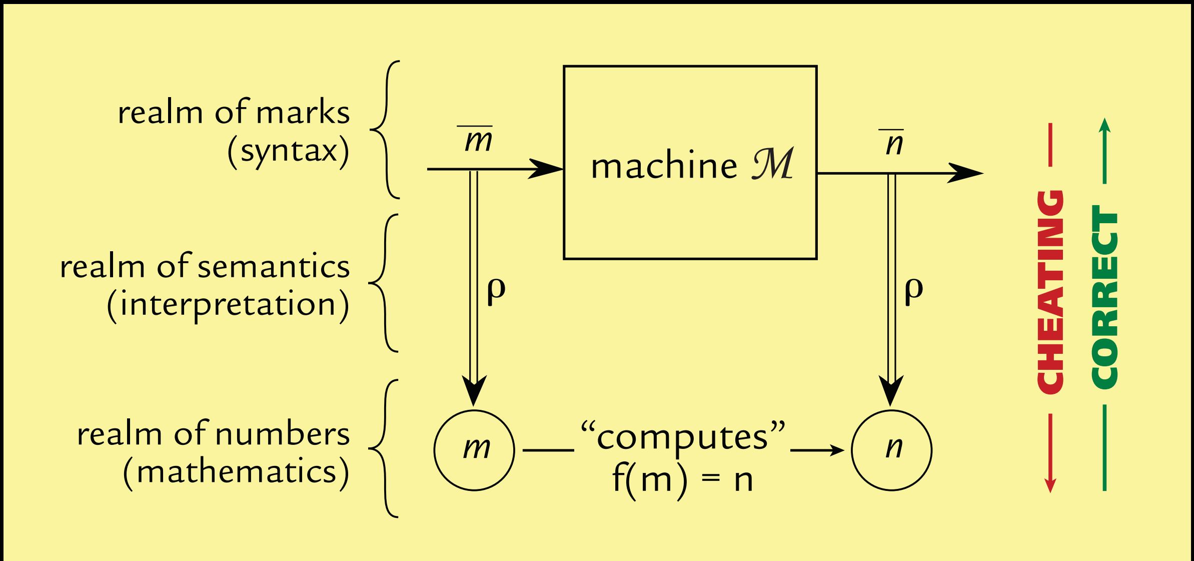
a. **TYPE 0** (the upper path) to be "represents 0", and

b. **TYPE 1** (the lower path) to be "represents 1"

5. Then the situation in the figure is perfectly well satisfied.
6. Moral: if we are allowed to individuate paths semantically, then the three criteria (C1′–C3′) don't do any work

Effectively different answer (types)

We haven't yet caught ★, but it is becoming clear why...



1. The intent of C1′–C3′ was to shift attention upwards

   a. Away from bottom and middle row issues (math and representation),

   b. Onto the upper row issue of the output's "**form**"

2. It has always been possible, through sufficiently devious means, to shift attention back down again, to semantics and represented mathematical entities

3. One might try to refine C1′–C3′, to prohibit reference to relations of semantics and interpretation—but this is almost guaranteed never to work

   a. One would need to know what ρ was, in order to prohibit access to it—and prohibiting access to a function is mighty hard...

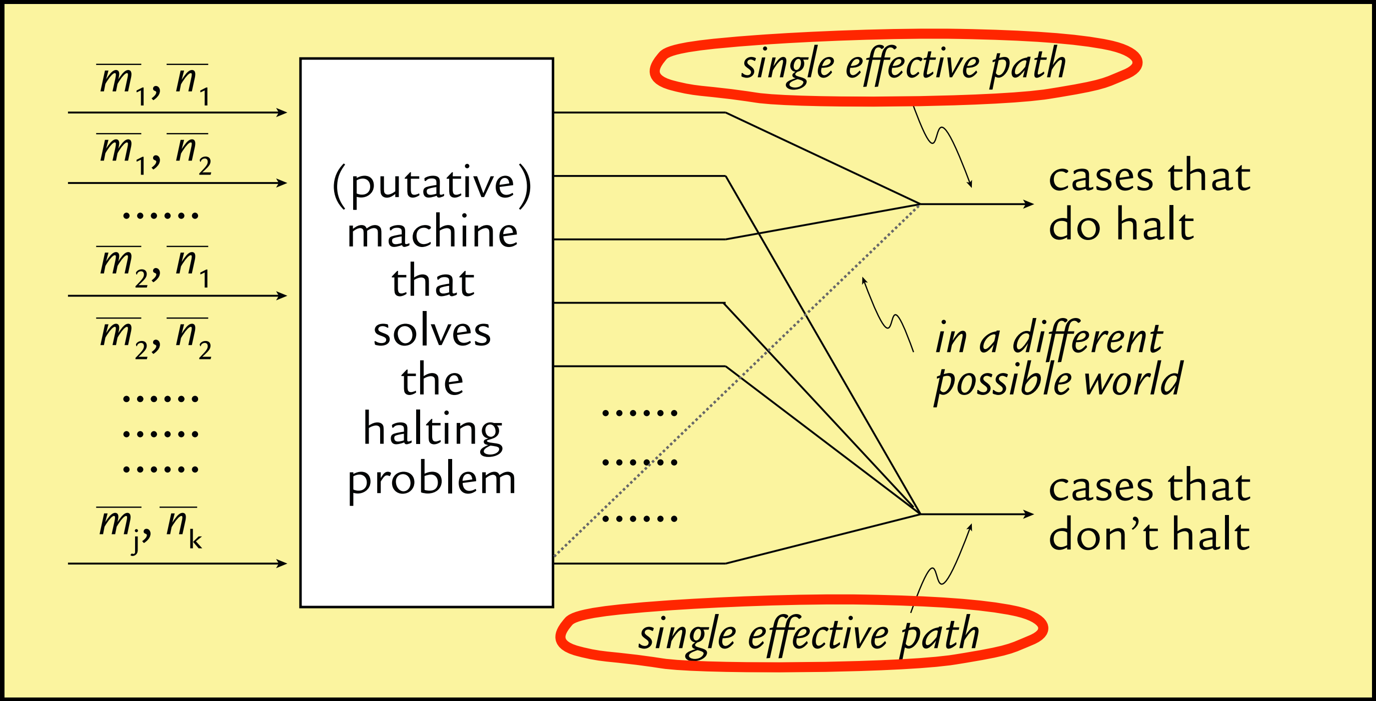
4. A better strategy: instead of try to rule out what is illegitimate, we need to be more constructive, and rule in what is legitimate

Effective paths

Try to formulate a criterion that forces the “tokens of the same type” to be similar at the level of form (not semantically).

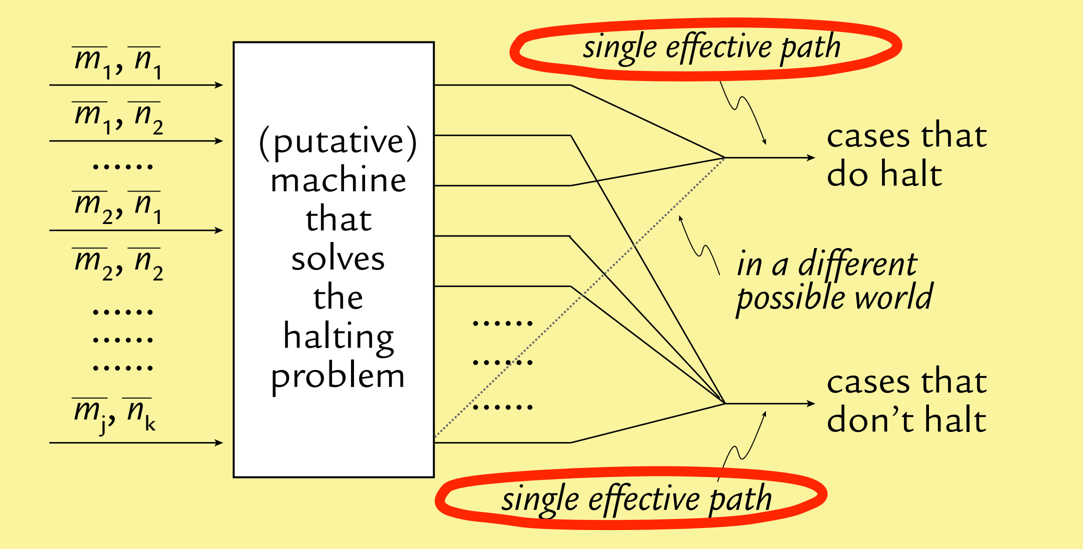
C4′ — Marshal all inputs that represent situations where machines halt (i.e., of **TYPE 0**) onto one effective path, and all inputs that represents situations where machines do not halt (i.e., of **TYPE 1**) onto a different effective path.

What’s new and different about this is the inclusion of the predicate ‘effective’


$\overline{m}_1, \overline{n}_1$
$\overline{m}_1, \overline{n}_2$
......
$\overline{m}_2, \overline{n}_1$
$\overline{m}_2, \overline{n}_2$
......
......
......
$\overline{m}_j, \overline{n}_k$
(putative) machine that solves the halting problem
......
......
......
single effective path
cases that do halt
in a different possible world
cases that don't halt
single effective path

Effective paths (cont'd)

1. We don't yet have a definition of a "single effective path," but it should at least require that 3 things can be done immediately and effectively
   a. Differentiate all 0s from all 1s
   b. Unify all 0s
   c. Unify all 1s



$\overline{m}_1, \overline{n}_1$

$\overline{m}_1, \overline{n}_2$

......

$\overline{m}_2, \overline{n}_1$

$\overline{m}_2, \overline{n}_2$

......

......

......

$\overline{m}_j, \overline{n}_k$

(putative) machine that solves the halting problem

......

......

......

single effective path

cases that do halt

in a different possible world

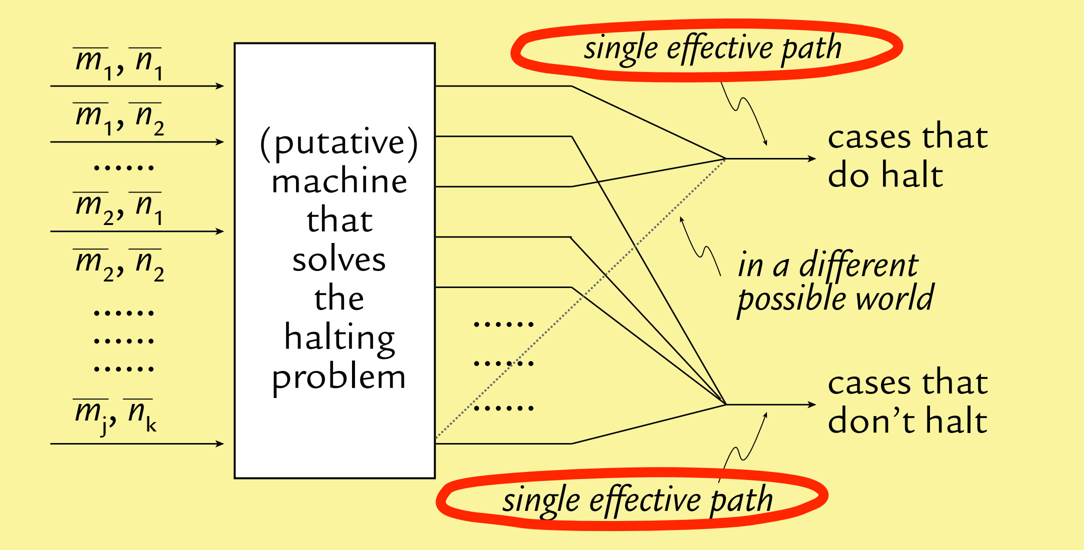
cases that don't halt

single effective path

2. Assemble this into a final problem statement:

H7 – Given as input marks $\overline{m}$ and $\overline{n}$, representing numbers m and n, respectively, produce as output tokens of mark types $\overline{0}$ or $\overline{1}$, representing numbers 0 or 1, respectively, depending on whether the Turing machine $\mathcal{M}$ (indicated by $\overline{m}$) would or would not halt if given $\overline{n}$ as input, such that (i) all tokens of $\overline{0}$ lead (immediately) to a single effective state or path, (ii) all tokens of $\overline{1}$ lead (immediately) to a single effective state or path, and (iii) all tokens of $\overline{0}$ are (immediately) effectively discriminable from all tokens of $\overline{1}$.

★ — Caught!



$\overline{m}_1, \overline{n}_1$
$\overline{m}_1, \overline{n}_2$
......
$\overline{m}_2, \overline{n}_1$
$\overline{m}_2, \overline{n}_2$
......
......
......
$\overline{m}_j, \overline{n}_k$
(putative) machine that solves the halting problem
......
......
......
single effective path
cases that do halt
in a different possible world
cases that don't halt
single effective path

1. Sure enough, H7 works

✔ *H7 catches* ★

2. But are we done?

No!

3. There are still a host of problems!

Problems with C4 / H7

1. Circularity:
   a. C4 is defined in terms of "effectiveness"—the very notion we are trying to provide a definition of!
   b. This is a very serious issue
   c. It means that we haven't answered the first horizontal question: $Q \cdot \varepsilon 1$
2. Encoding (ρ)
   a. We got here by worrying about encoding—but C4/H7 do not mention encoding at all!
   b. That means that we haven't answered any vertical questions, either! ($Q \cdot \rho 1$ through $Q \cdot \rho 3$)
3. That is: we *haven't answered any of our initial questions at all!*
4. And in case one were to think this is all about the halting problem, note that every one of these concerns applies to all instances of computation...

Multiplication

Task: construct a machine $\mathcal{M}^*$ to "multiply numbers"

1. As with H, we cannot define $\mathcal{M}^*$ purely in terms of functions, arguments, and values (FAVs)—requiring it, for example, given as input numbers m_1 and m_2, to produce as output $m = m_1 \cdot m_2$

   MULT1 — Given as input numbers m_1 and m_2, produce as output number m, such that $m = m_1 \cdot m_2$

2. Nor is it enough to put in a minimal representational requirement:

   MULT2 – Given as input marks $\overline{m}_1$ and $\overline{m}_2$, representing numbers m_1 and m_2, respectively, produce as output mark $\overline{m}$, representing $m_1 \cdot m_2$

3. MULT2 could be satisfied by a machine as vacuous as ★ — in particular, by one that simply prints out "$m_1 \cdot m_2$"
4. To "force" $m_1 \cdot m_2$ to actually be computed, we need to use the same strategy we used for H7

Multiplication (cont'd)

1. An effective specification of multiplication:

MULT3 – Given as input marks $\overline{m_1}$ and $\overline{m_2}$, representing numbers m_1 and m_2, respectively, produce as output mark $\overline{m}$, representing $m_1 \cdot m_2$, such that: (i) for any given (product) number m, all tokens of $\overline{m}$ lead (immediately) to a single effective path or state; and (ii) for any j and k, if $m_j \neq m_k$, then the effective path of $\overline{m_j}$ is effectively discriminable from the effective path of $\overline{m_k}$

2. There is an interesting difference between H7 and MULT3

a. In the case of H7—which finally stated the halting problem in an acceptable way—there was **no role for representation (ρ) at all**.

b. In the case of MULT3, there *is* reliance on representation (ρ)—because it is only through the marks representing numbers that the operation can **held to account as multiplication**.

3. We will get back to this!

Morals so far

1. Meantime we have some more results

R4: No requirement formulated at the level of functions, arguments, and values can ever force a function to be "computed"

R5: Computing a function does not "happen" at the level of FAVs. It happens at the level at which the functions, arguments, and values are *represented*

Yet note R5 doesn't mean that computation cannot be understood at the level of FAVs.

a. The traditional semantical approach, which we saw in logic, may still apply—it is certainly **essential** to MULT3.

b. Yet if it does apply, then why did ρ need to be so severely constrained?

c. This just brings us back to the "vertical" questions Q · ρ1— Q · ρ3

d. None of these have been answered yet!

Uninterpreted marks

Look again at criterion **C4′**—the crucial step in catching ★. (And keep in mind that our entire focus in analysing H was on the **output**; we could have, but did not, raise similar concerns about the **inputs**).

> **C4′**— Marshal all inputs that represent situations where machines halt (**TYPE 0**) onto one effective path, and all inputs that represents situations where machines do not halt (**TYPE 1**) onto a different effective path.

Not only does **C4′** place no **demands** on issues of representation; it doesn't even require that the ***outputs* represent anything at all**

All it required was that the outputs be effectively discriminable

Similarly, H7 did not require the output to be representational, either. That is:

> R6: The only tenable formulation of the halting problem that we were able to come up with *eliminated all representational requirements on the outputs entirely*

At least as regards the **halting problem**, it starts to look as if the whole business about representation may be **a snare and a delusion**

Argument so far

1. In our effort to catch ★, we started out on the bottom row, considering mathematical entities — and failed
2. We then moved (upwards), to the middle row, to considering representation and encoding (ρ). At first that seemed to help, but eventually ρ snuck back in
3. So we stopped looking at ρ, and focused on marks alone (not on mathematics or semantics at all)
4. That is (with respect to the output), we:
    a. Started at the bottom level (numbers, realm of mathematics)
    b. Then included the middle level (semantics, realm of interpretation)
    c. Then included the upper level (marks, realm of syntax)
    d. Then threw away the bottom two levels
5. What about ρ?
    a. Started out very concerned about ρ
    b. Figured out that ρ must be very constrained (which was interesting)
    c. Now ρ has disappeared!

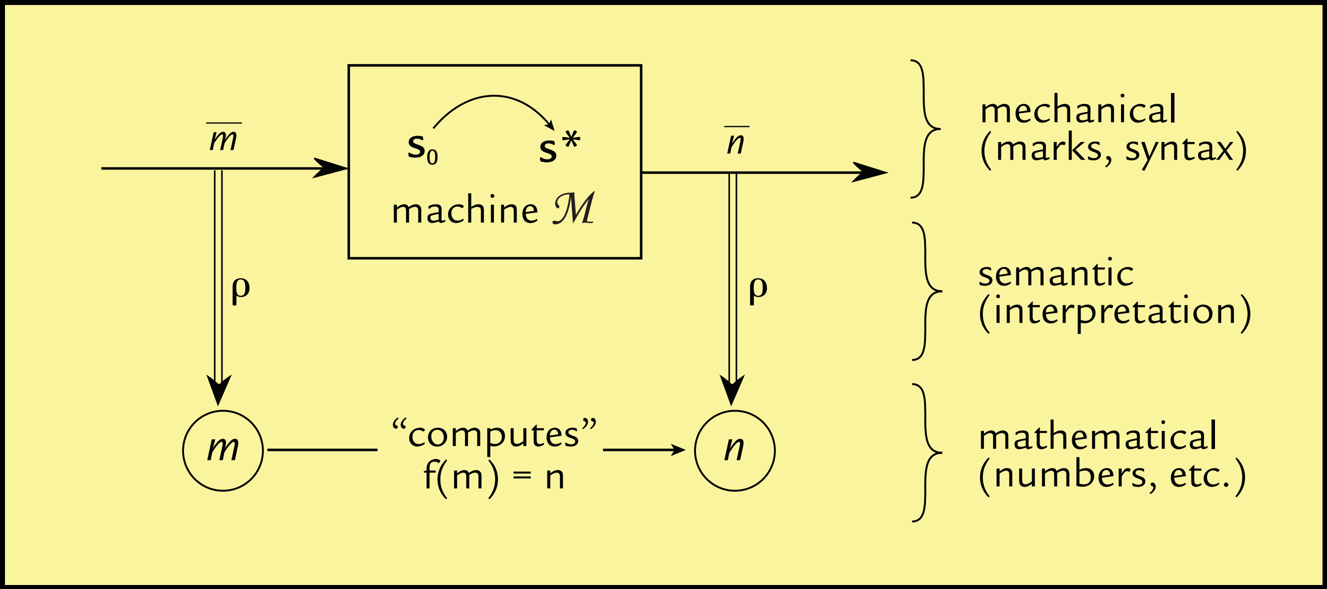
Halting problem revisited

1. What do we conclude, from the fact that ρ was irrelevant in our example?
2. We said at the outset that the outputs needn't be 0 or 1.
3. Now we've concluded that they needn't be representational at all.
4. So 0 and 1 are off the table.
5. All we have are two "discrimimable effective paths".
6. Why not use standard convention for classifying binary oppositions, and associate the two effective paths with the numbers 0 and 1 (which are free for use, now)
7. This opens the door for the final (obvious?) insight

R7: The marks $\overline{0}$ and $\overline{1}$ do not represent the numbers 0 and 1, after all.

Rather, the *numbers 0 and 1* represent (or at least model) the *marks $\overline{0}$ and $\overline{1}$!*

Turning the representation relation upside-down!



$\overline{m}$

s_0 s^*

machine $\mathcal{M}$

$\overline{n}$

ρ

ρ

m

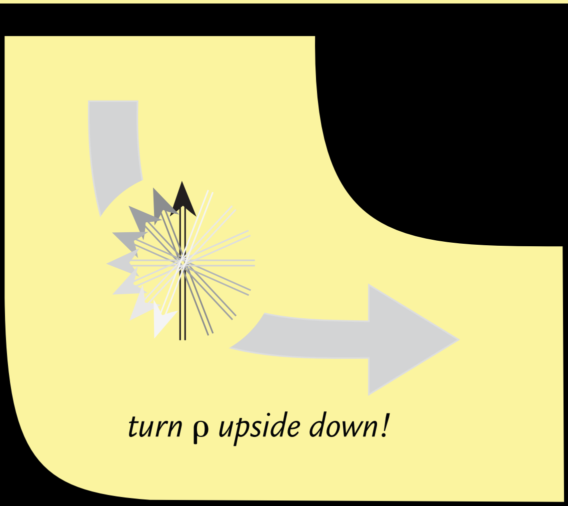
“computes”
f(m) = n

n

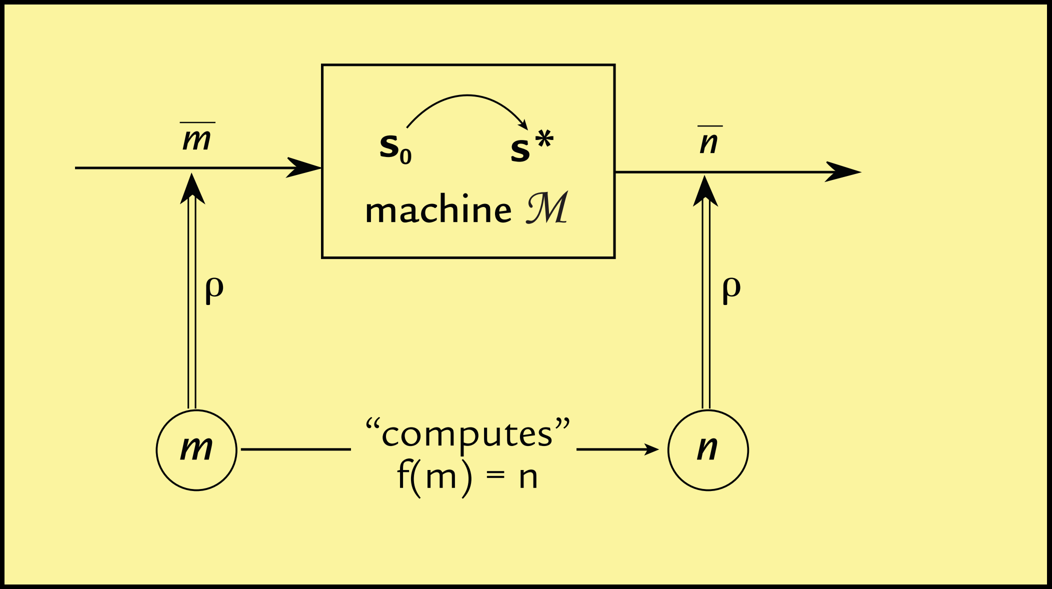
mechanical
(marks, syntax)

semantic
(interpretation)

mathematical
(numbers, etc.)



turn ρ upside down!



$\overline{m}$

s_0 s^*

machine $\mathcal{M}$

$\overline{n}$

ρ

ρ

m

“computes”
f(m) = n

n

Contents

1. Context
2. Turing machines
3. The halting problem (H)
4. Solving H
5. Morals

Start climbing back up...

R8: In the mathematical theory of computation & computability, the representation relation ρ runs the other way around—from numbers to marks instead of from marks to numbers!

1. What we’ve concluded:

2. R8 explains a myriad things:

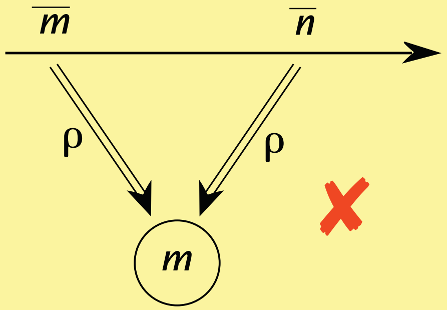
Q: Why did ρ need to be so simple?
A: So it can be inverted!

Q: Why is ρ never theorized?
A: Because it’s not part of the subject matter (it’s in the theoretician’s toolkit, like math & telescopes)

Q: Why does computing have a single subject matter (unlike logic, whose subject matter is double)?
A: Because it only studies marks and their manipulations

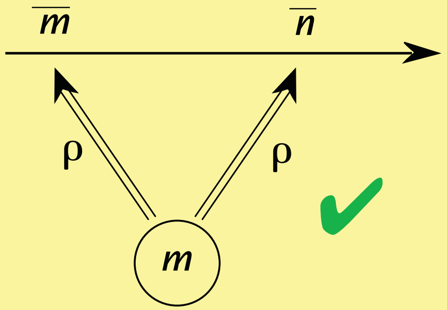
Co-reference (equality) was prohibited in “reasonable encodings”

Why?



Because ambiguity is prohibited in mathematical models

Obvious!



Theory of computation

1. Computability/complexity theory was never interested in representation
2. Rather, computability/complexity theory is a theory of marks
    a. Not: marks in all their glory
    b. Rather: the effective properties of marks
    c. Not: marks as written symbols
    d. Rather: marks at a higher level of abstraction (relevant to the arbitrary construction of machines
3. Leads to a strong, positive result

R9: In spite of the press, what has been called a theory of "effective computability" is in fact a *mathematical theory of the flow of effect*—that is: a *mathematical theory of causality!*

More answers to our questions

(Vertical) questions: about encoding (ρ)

$Q \cdot \rho 1$ — What are the constraints on “reasonable” encoding? ✓

$Q \cdot \rho 2$ — Why are they theoretically ignored? ✓

$Q \cdot \rho 3$ — Should they be studied (put back on the table)? ✗

(Horizontal) questions: about effectiveness (ε)

$Q \cdot \varepsilon 1$ — What is the fundamental character of the effectiveness (“computability”) constraints? ✓

$Q \cdot \varepsilon 2$ — Are these effectiveness constraints properties of

- The realm of marks (ε_p)? $\Longleftarrow$ Monday ✓
- The realm of semantics (ε_s)? $\Longleftarrow$ Saturday
- The realm of mathematics (ε_m)? $\Longleftarrow$ Last week

$Q \cdot \varepsilon 3$ — Can computability and complexity theory be conducted (as usual) at level of mathematics? ✗

Consequences for computing

Primarily, facts about computational *theory*

Some computational facts that this analysis explains:

1. Why complexity results (log, cubic, etc.) are defined over the lengths of the inputs (as marks)—not the mathematical magnitude of the arguments
2. Both the structure and the popularity of linear logic, intuitionistic type theory, the popularity of constructive mathematics, etc.
   a. Cf. Girard's "geometry of interaction"
   b. Linear logic looks utterly perverse, from a traditional logical point of view (i.e., within the model of logic we examined at the beginning).
3. The popularity of term models (e.g., in semantics for Prolog)
4. Why we have programming language semantics, not program semantics
5. Attempts to fuse computing, quantum mechanics, and mechanical models of information
   a. E.g., amount of heat associated with loss of one bit of "information"

Consequences for computing (cont'd)

6. What does this all mean for the overarching project: figuring out what computing is?
    a. A theory of causality is essentially a theory of mechanism
    b. But computing (as I've said since the beginning) is not just mechanism
    c. Computing involves an interplay of meaning and mechanism
7. This leads to the main negative result

R10: The "theory of effective computability" is not a theory of computation at all!

Assessment

Are these results as depressing, for computer science?

1. No, they are not indictments
2. On the contrary, a theory of what can be done effectively (mechanically) in this world—what it takes, how hard it is to do it, etc.—is an enormous achievement (worth a passel of Nobel prizes)
3. It is just that what has been called the "mathematical theory of computation" or the "theory of effective computability" is in fact a theory of only one aspect of computing: the "mechanism" half.
4. It will apply to computing (though it will also apply to lots of things are not genuinely "computational")

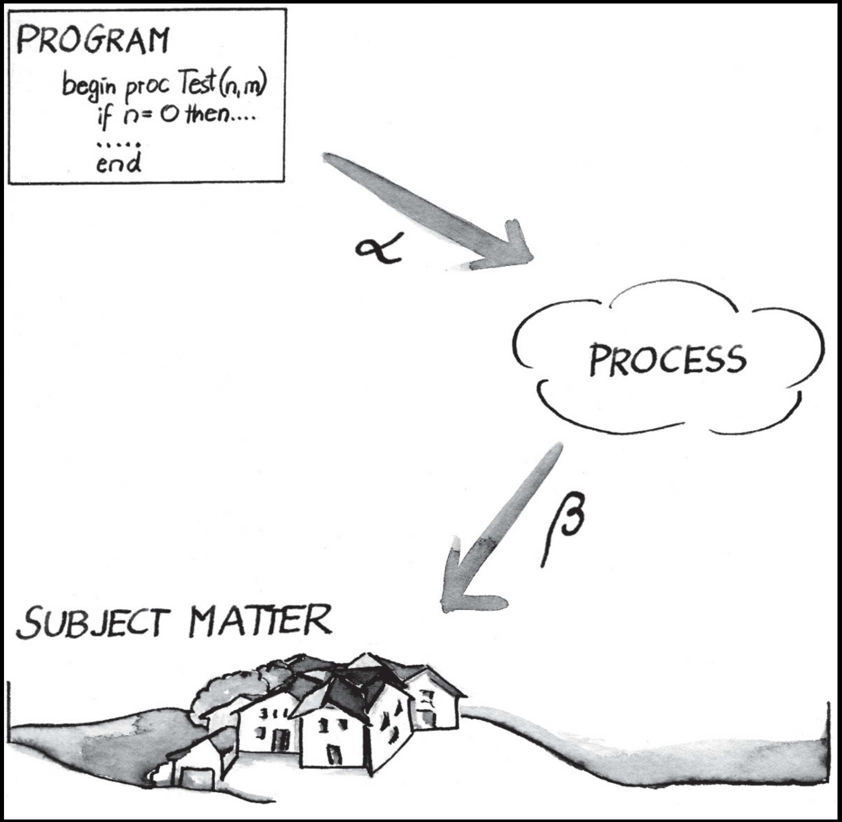
R11: The reconstruction of the so-called "theory of computability" as a mathematical theory of the *flow of effect*—i.e., a mathematical *theory of causality*, analysing the powers and limitations of mechanism—does not render it irrelevant to a comprehensive theory of computation. It is just not a theory of computational systems *as computational*.

Assessment (cont'd)

What do we need if we are to have a full theory of computation—one that recognizes it as genuinely involving both meaning and mechanism?

1. The fundamental issue has to do with meaning and semantics
2. Computation *is* semantical. In that sense the original analogy with logic was correct.
3. The problem is that the role of semantical interpretation (ρ) in analysing Turing machine tapes was distracting.
4. At first it looked as if it was there to deal with computation's semantical aspect.
5. But then we realized that was wrong. In the theory of "effective computability," it was playing a meta-theoretic role, for classifying mechanical effective states. So the semantical aspect never got recognized—or analyzed.
6. So what a full theory of computing needs is a theory of genuine semantics (relations between computational processes and the world—as in logic).

Programming language semantics



1. You might think that there *is* a place where semantics is analysed in computer science: in the area of programming language semantics (axiomatic, denotational, operational, etc.)
2. In separate work, I argue that that this PL work also fails to analyse the true nature of computational semantics
3. Specifically, it focuses on the "program-process" relation (α in the figure), whereas what we need is an analysis of the relation between process and world (β).
4. Note that the "process-world" relation (β) will in general not be effective—it is not an activity or process, not something that happens in time, not "computable"
5. β is the computer version of what (in R2) I called the "human interpretation function" (ρ′)—vastly more powerful than the original or "inverted" sense of ρ.
6. It is this β (ρ′) that we need to analyse, for a full theory of computing…

Assessment (cont'd)

1. In spite of its benefits, the costs of this analysis are dire
2. It will involve reconstructing
   a. Complexity results (e.g., prime factorization—can't be characterized as that!)
   b. Absolute computability results (including halting problem!)
3. Status
   a. The "theory of computability" is currently framed at "bottom level"
   b. My claim: that formulation is wrong
   c. Required: to reconstruct all these results as stories at the top level—about:
      i. The effective structure of machines
      ii. The effective structure of inputs
      iii. The effective structure of outputs
4. Example: I am involved in reconstructing the Church-Turing thesis as something I call a "motor thesis"—along something liek the following lines:
   a. With a sufficient set of passive, perfect parts (no friction, etc.), plus one active source (a "motor"), you can construct a mechanism whose overall external behaviour, modulo issues of geometric time, is isomorphic to the overall external behaviour of any physically constructable machine.

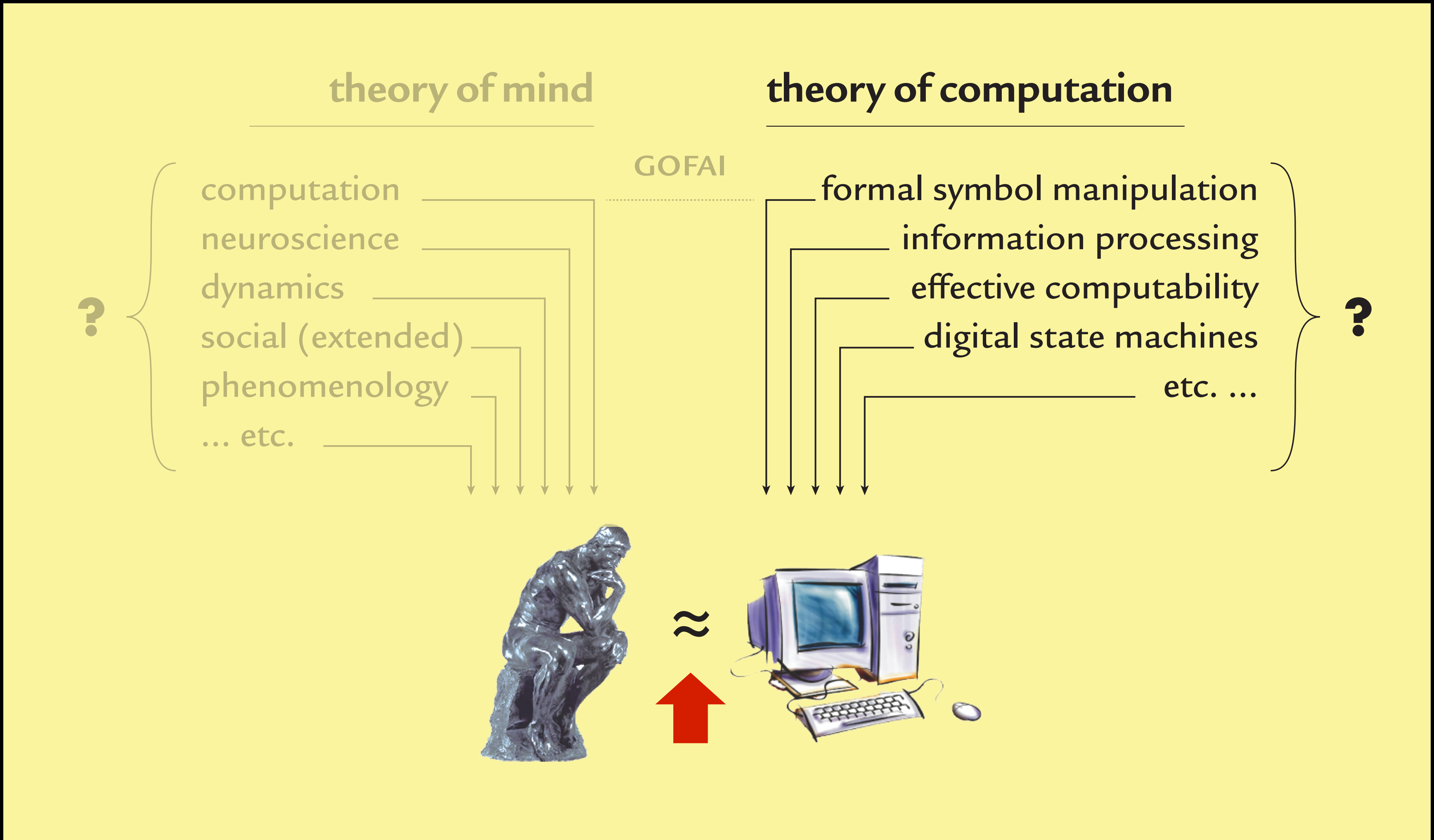
Assessment (cont'd)

5. And so on ...

There is a huge amount of homework to do!

Postscript on cognitive science

1. What about the computational theory of mind?



theory of mind

?

computation

neuroscience

dynamics

social (extended)

phenomenology

… etc.

GOFAI

theory of computation

formal symbol manipulation

information processing

effective computability

digital state machines

etc. …

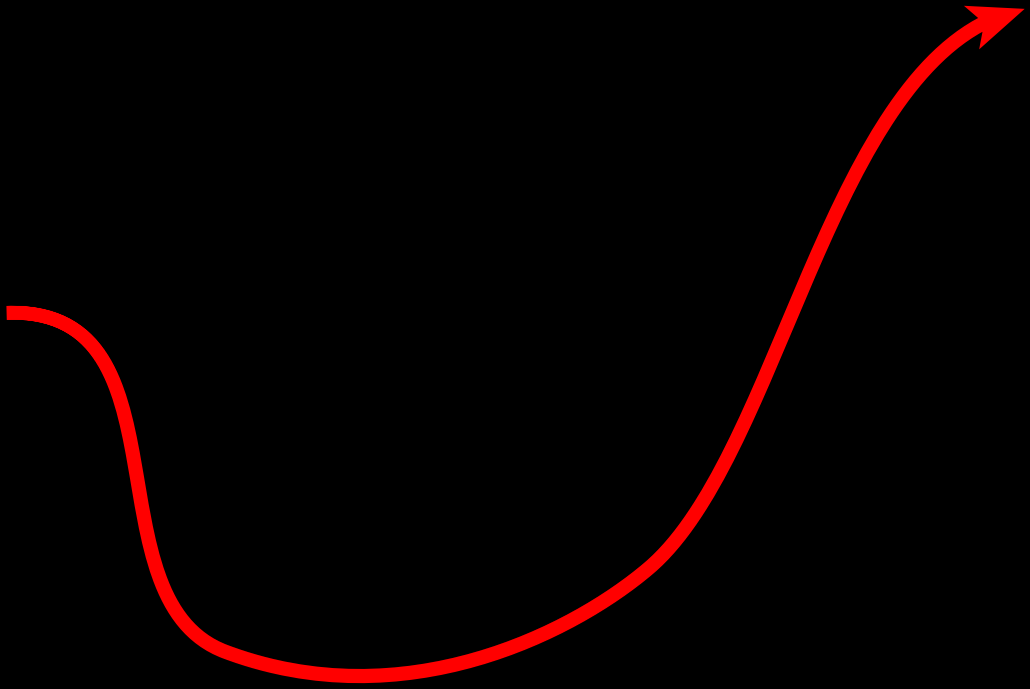
?

≈

Postscript on cognitive science (cont'd)

2. What our analysis has shown is that the only constraints on computing derive from fundamental constraints of mechanism or materiality
3. I claimed that that mechanisms isn't all there is to computing—that (at least in my view) computing involves an interplay of ***meaning*** and mechanism
4. What our analysis shows, though, is that, because effectiveness (ε) amounts to nothing more than what is physically/causally possible (ε_P), there are no constraints on being computational besides being a material meaningful thing
5. So is the computational theory of mind true?
6. Minds are surely loci of meaning—what could meaning be, if not something that minds mess with?
7. So the only real question (if it is one) has to do with mechanism.
8. "Are we computers" is the same as "are we physically embodied creatures?"

R12: The only way to reject the computational theory of mind is to be an outright dualist. As long as dualism is false, therefore, the computational theory of mind is true. Boring—but true.



Thanks