Causality, Self-reference and Self-referential Dynamics

Jiang Zhang

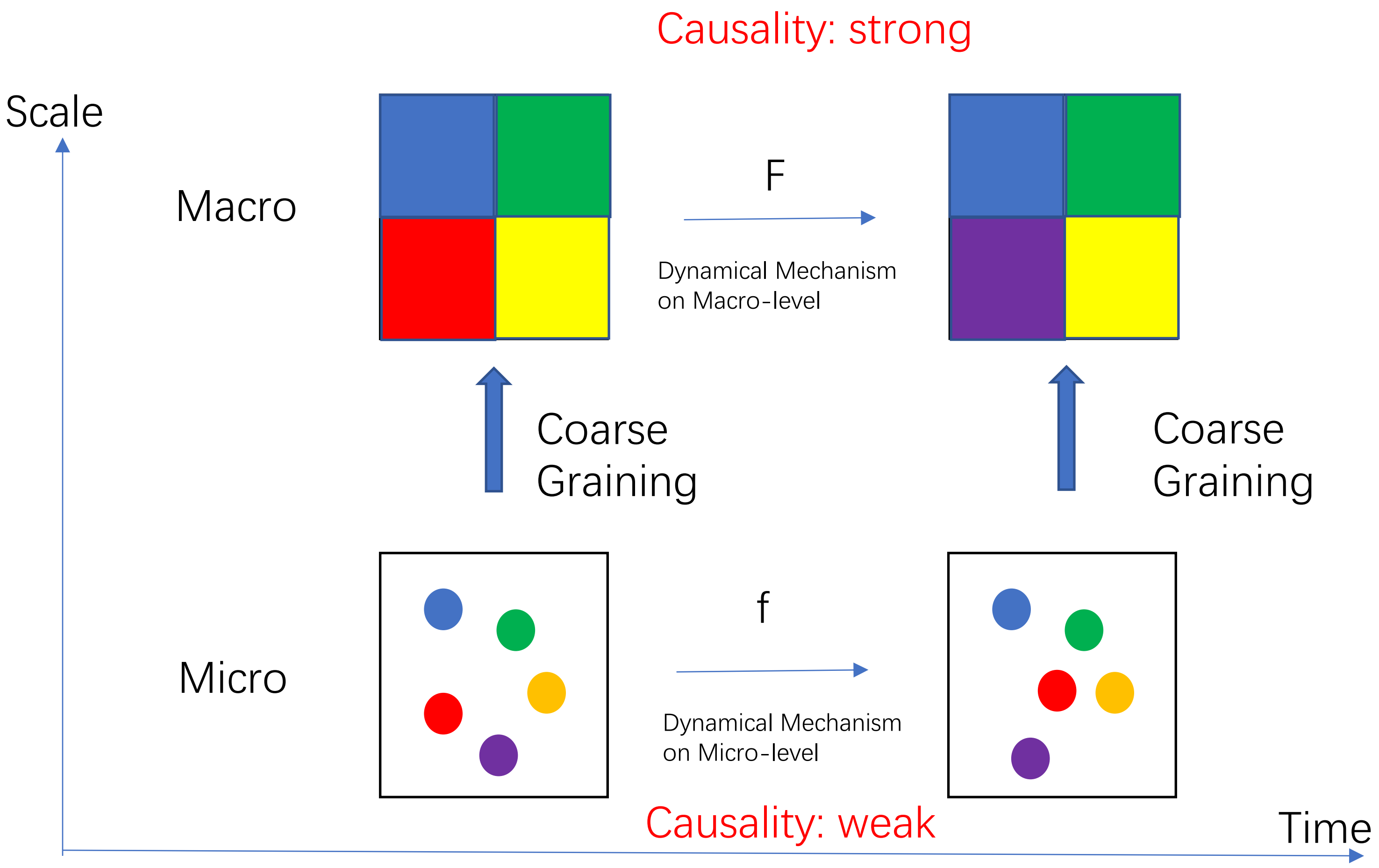
School of Systems Sciences, BNU

Swarma Club, Swarma Compus Ltd. Co.

Outline

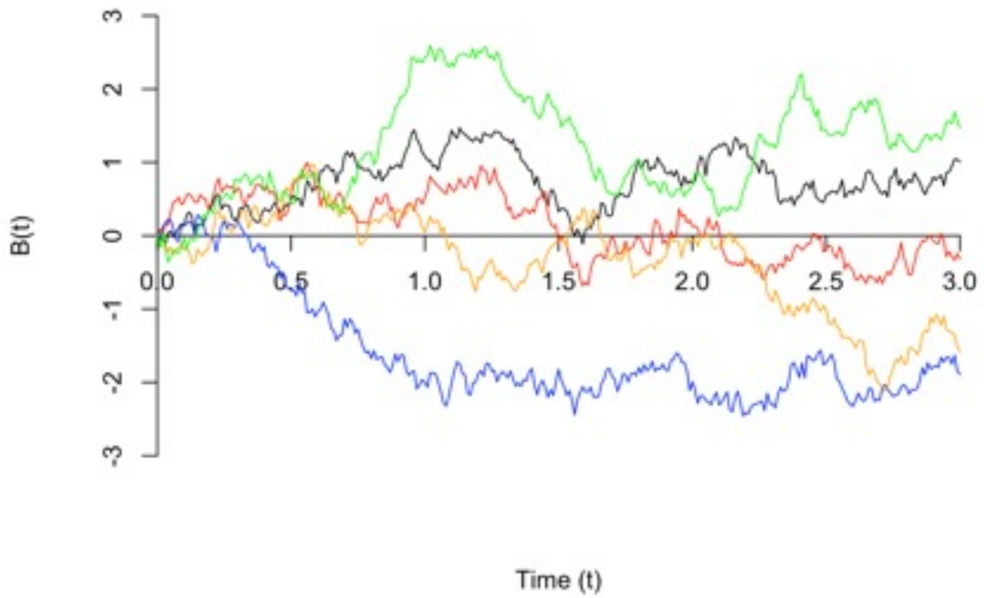
- Why Self-reference?
- Self-reference in Computational Theory
- Self-referential Dynamics

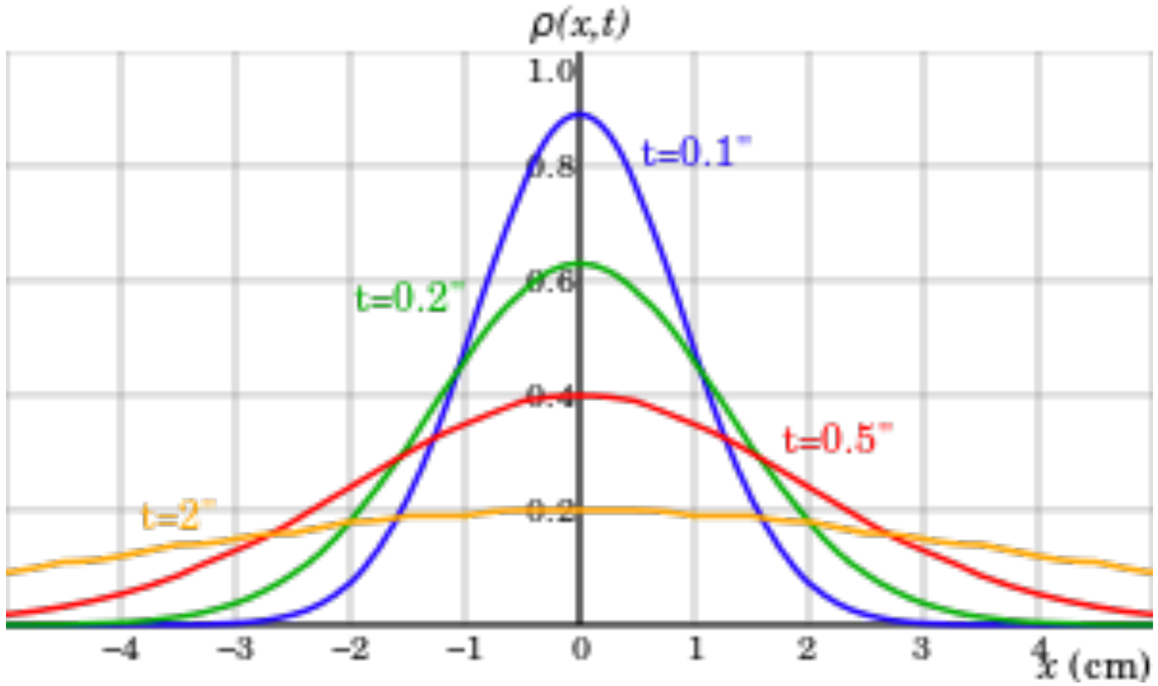
What is causal emergence



Causal emergence: example

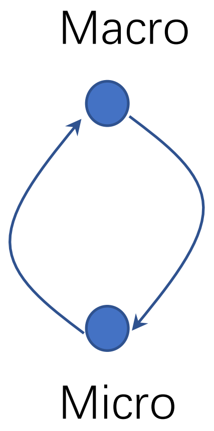
$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2}$$

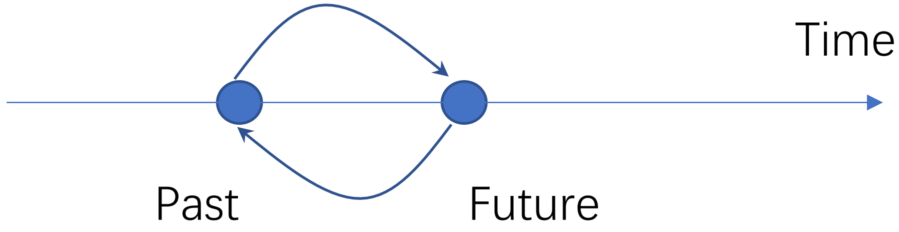




More difficult causal emergence?

- Emergence
  - Top-down causality
  - Formal cause of Aristotle
- Free will
  - Subjectivity
  - Final cause of Aristotle



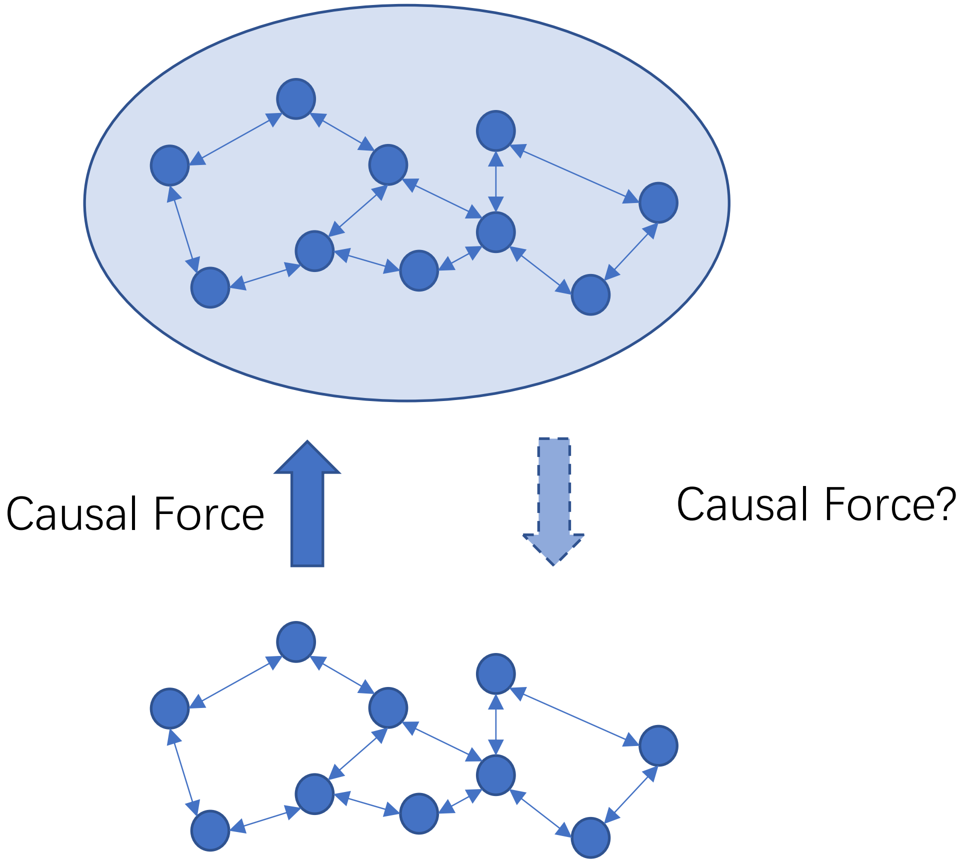


Top-down(formal) cause emergence

食蚁兽：有一天去拜访马姨（一个蚁群），碰到了一件事，……
螃蟹：快告诉我你们碰到了什么事了？
食蚁兽：马姨那天感到很孤独，很想同别人聊聊。因此，她很大方地要我自己挑那些我所能见到的最鲜嫩的蚂蚁。（她从不吝啬她的那些蚂蚁）

——候世达：《哥德尔、埃舍尔、巴赫》：蚂蚁赋格

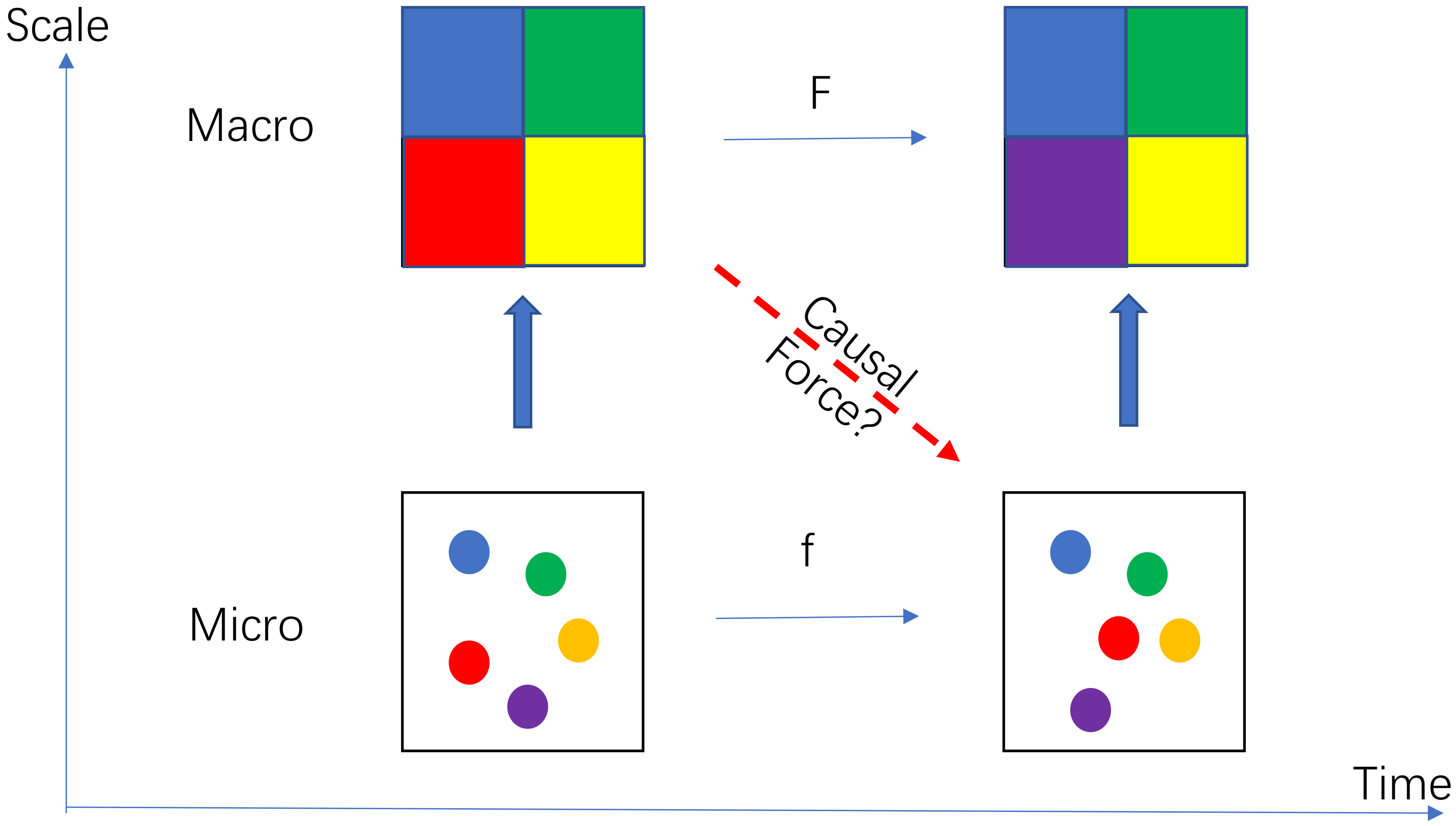
Feedback from top to down



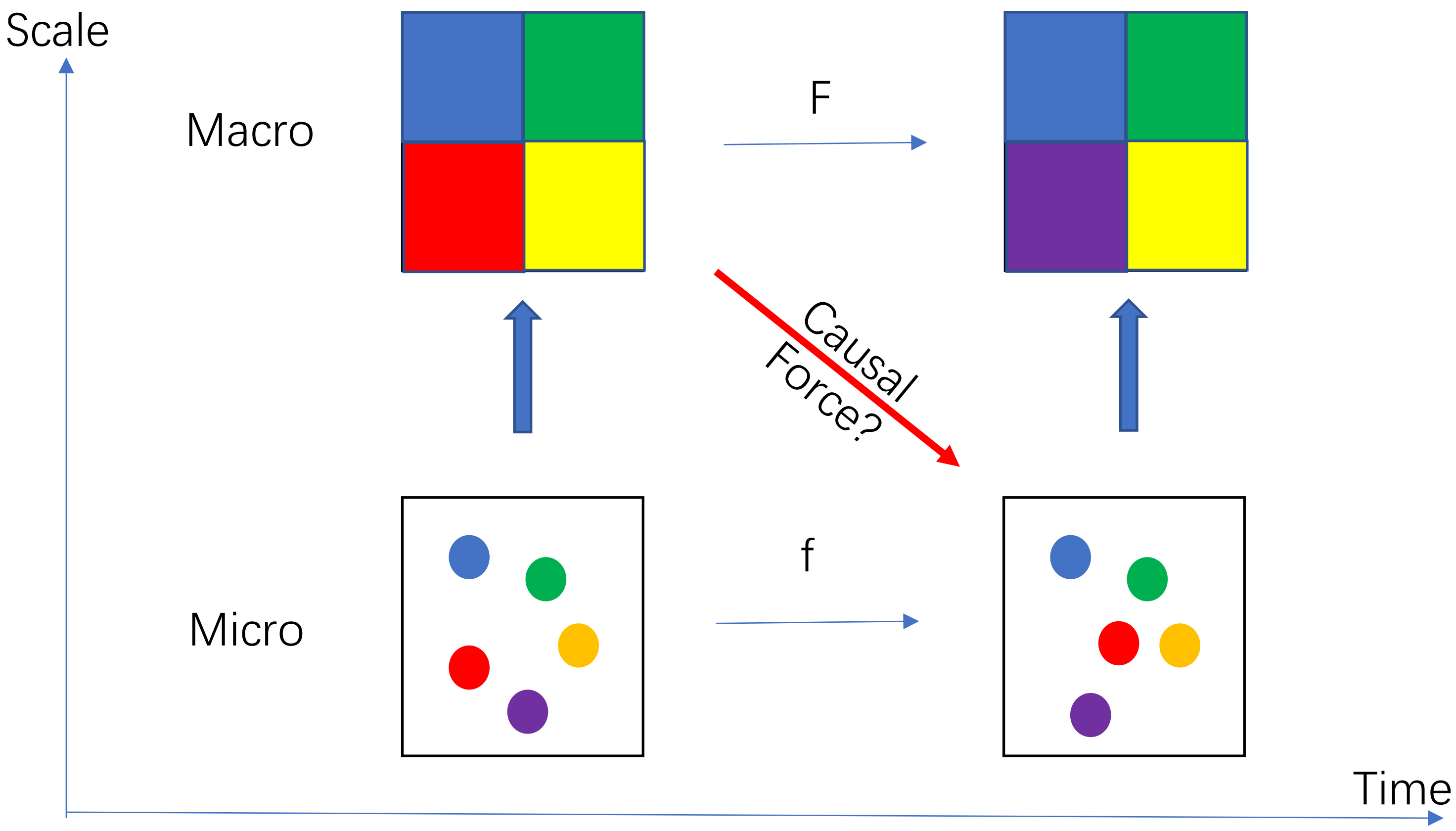
Causal Force

Causal Force?

New problem?

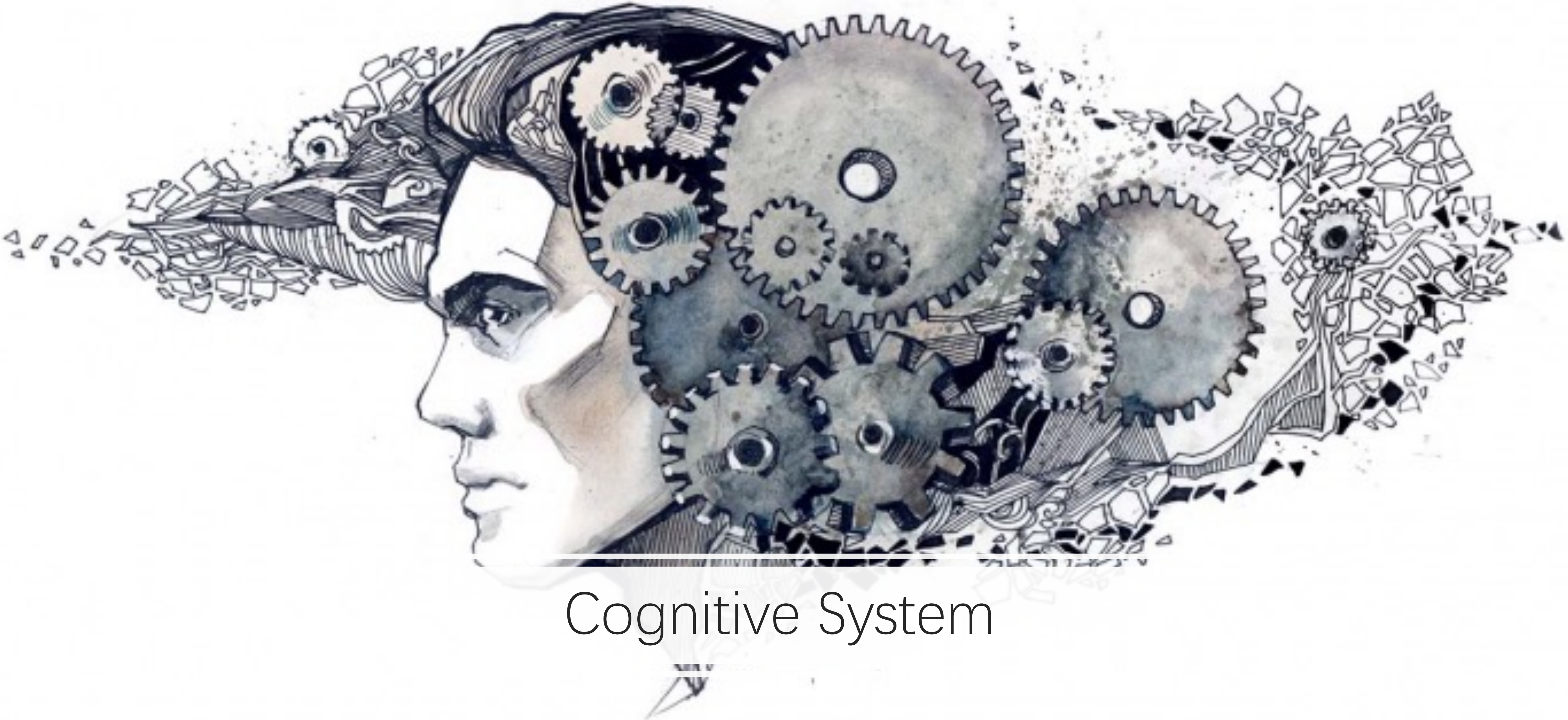


New problem?



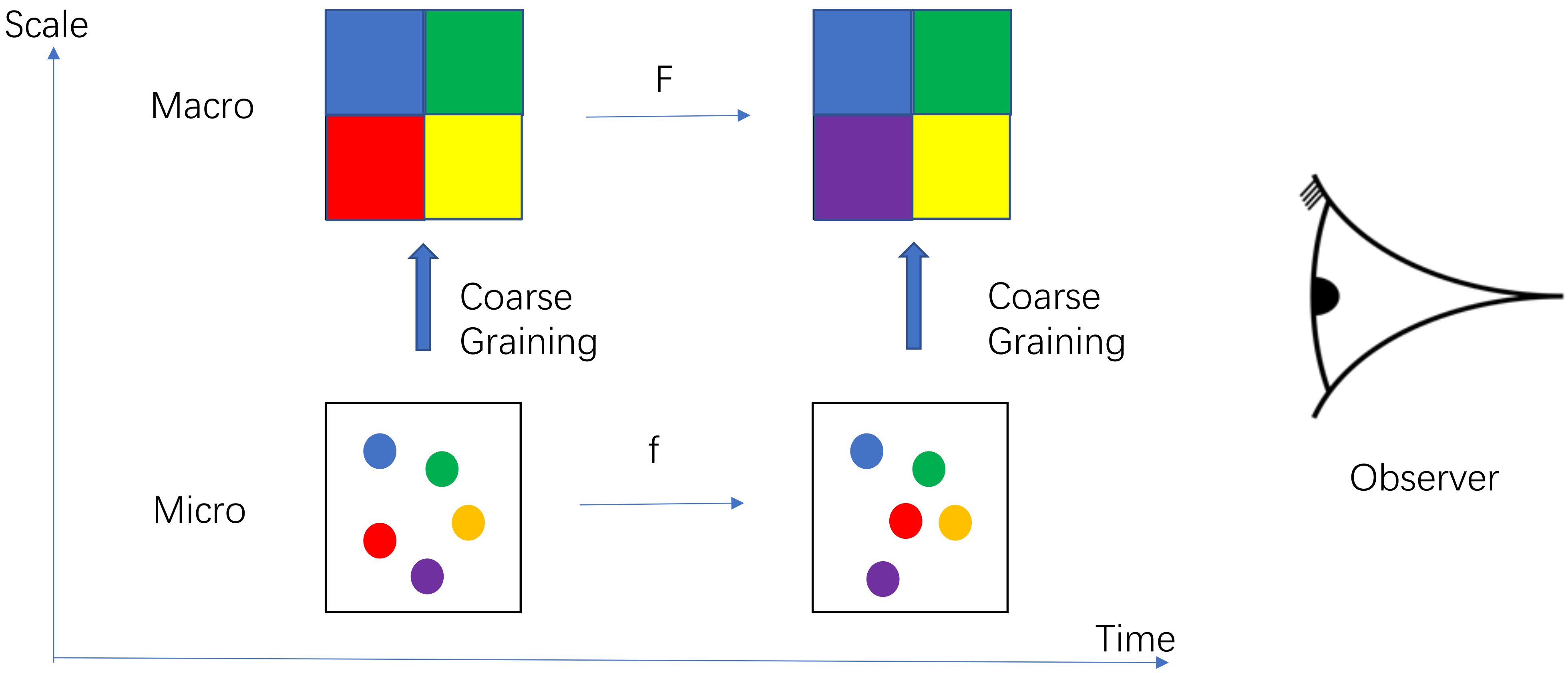
Example:

Stock Price, Statistics

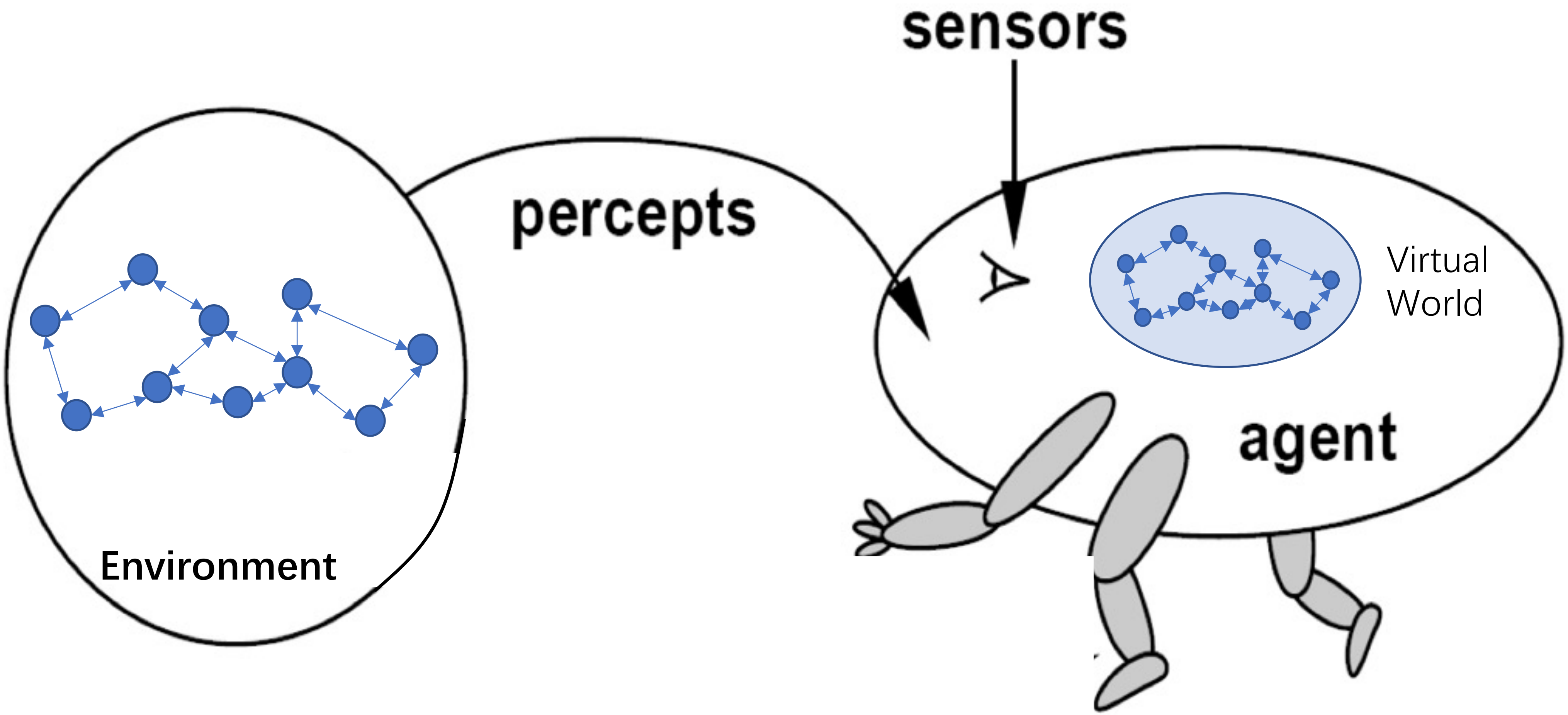


Cognitive System

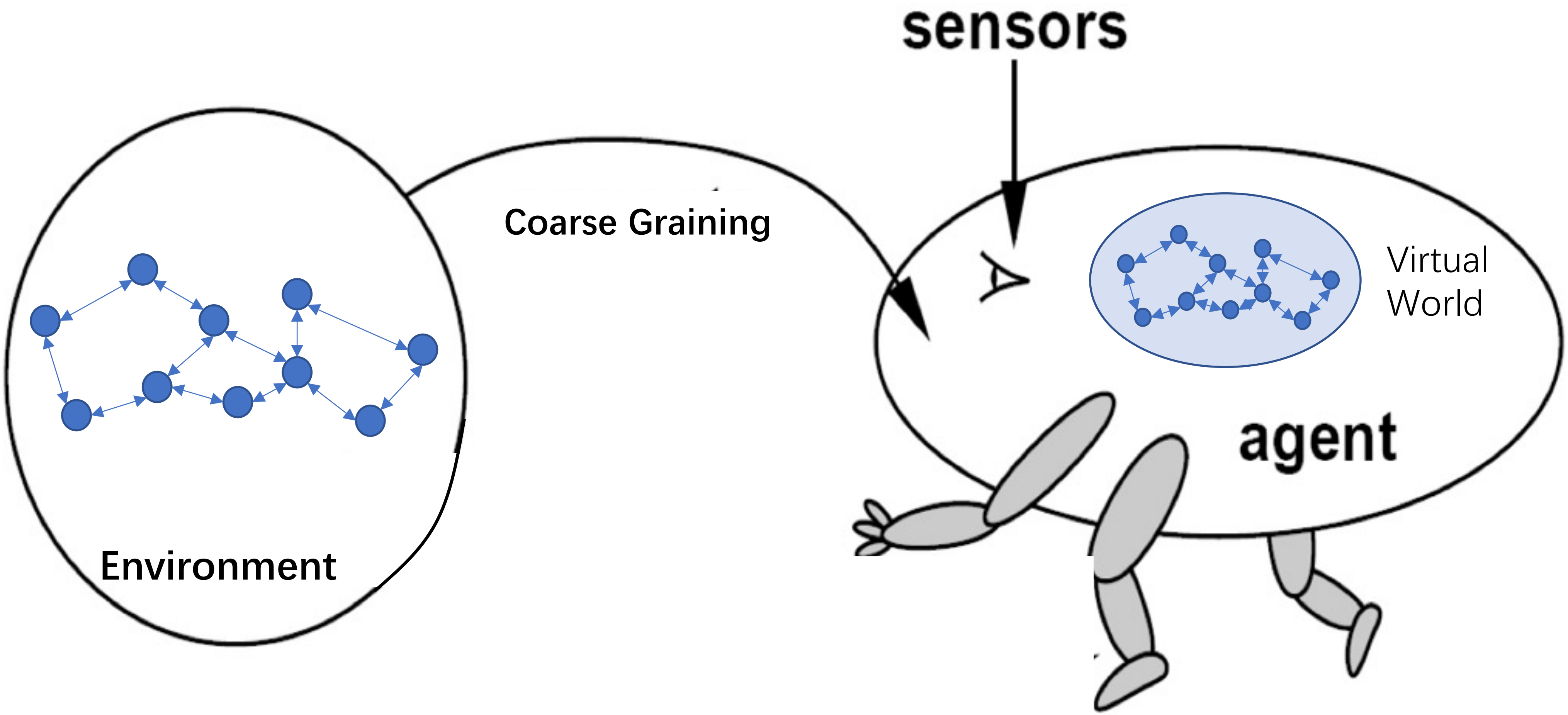
What is causal emergence



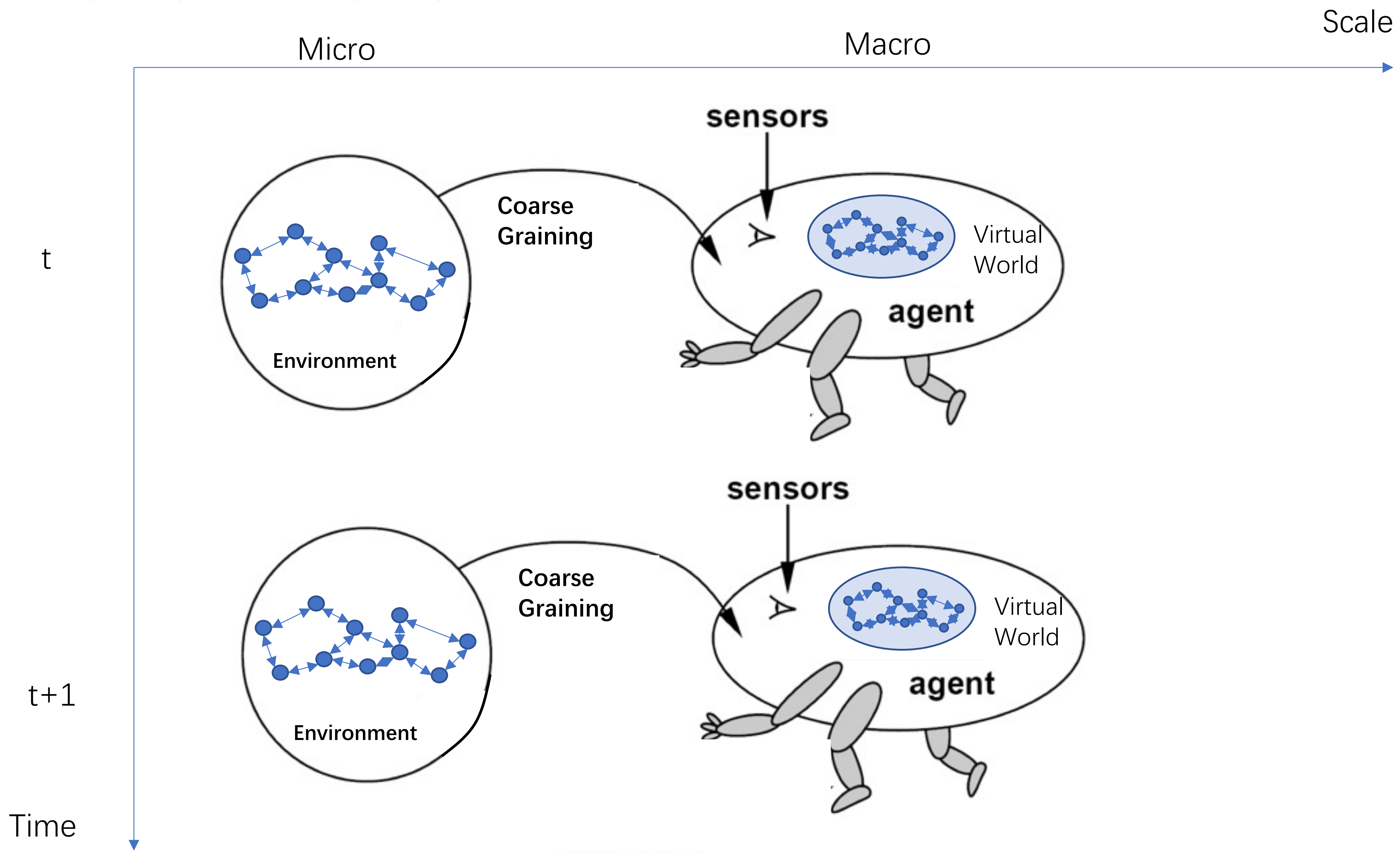
Cognitive System



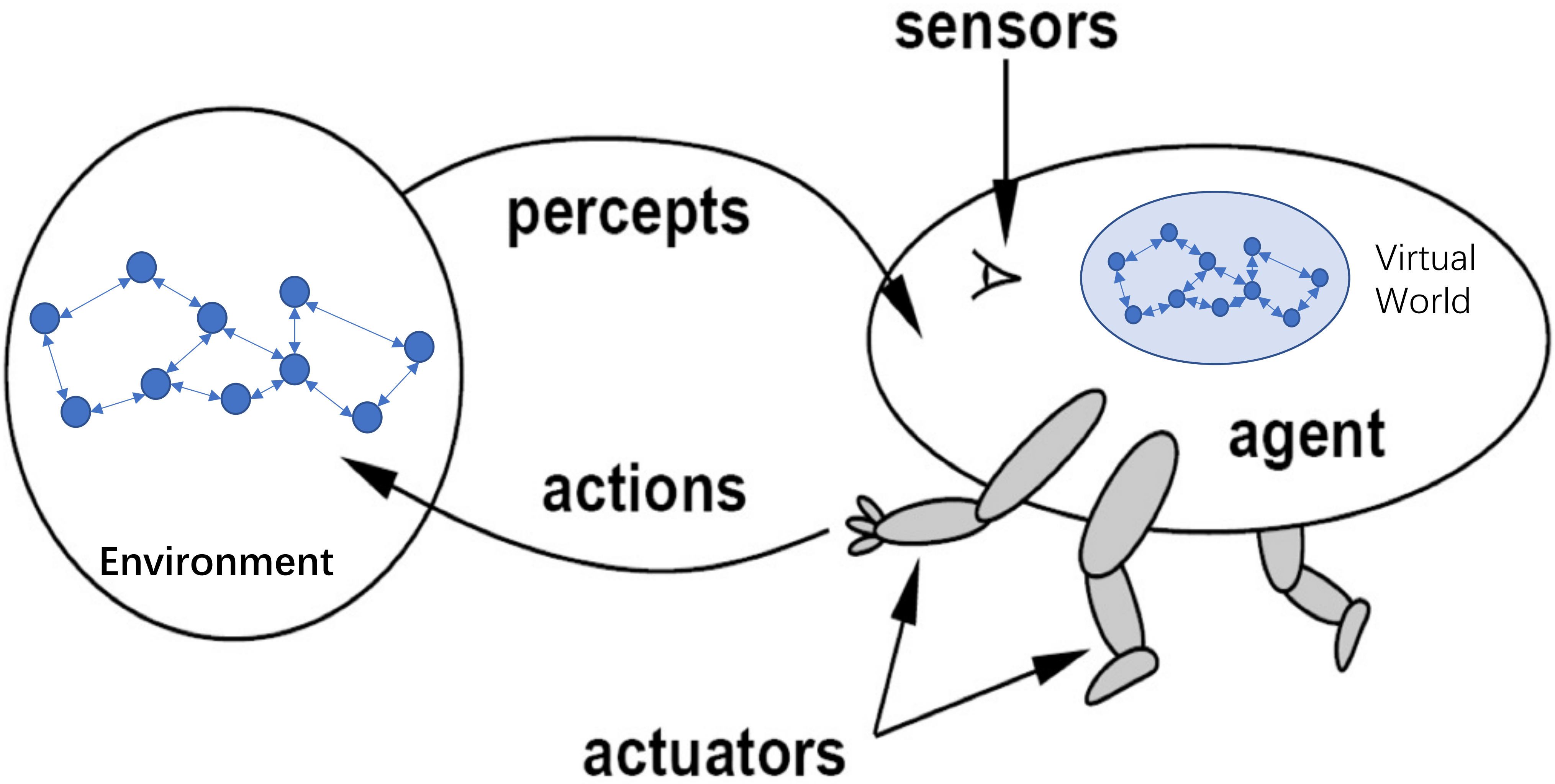
Cognitive System



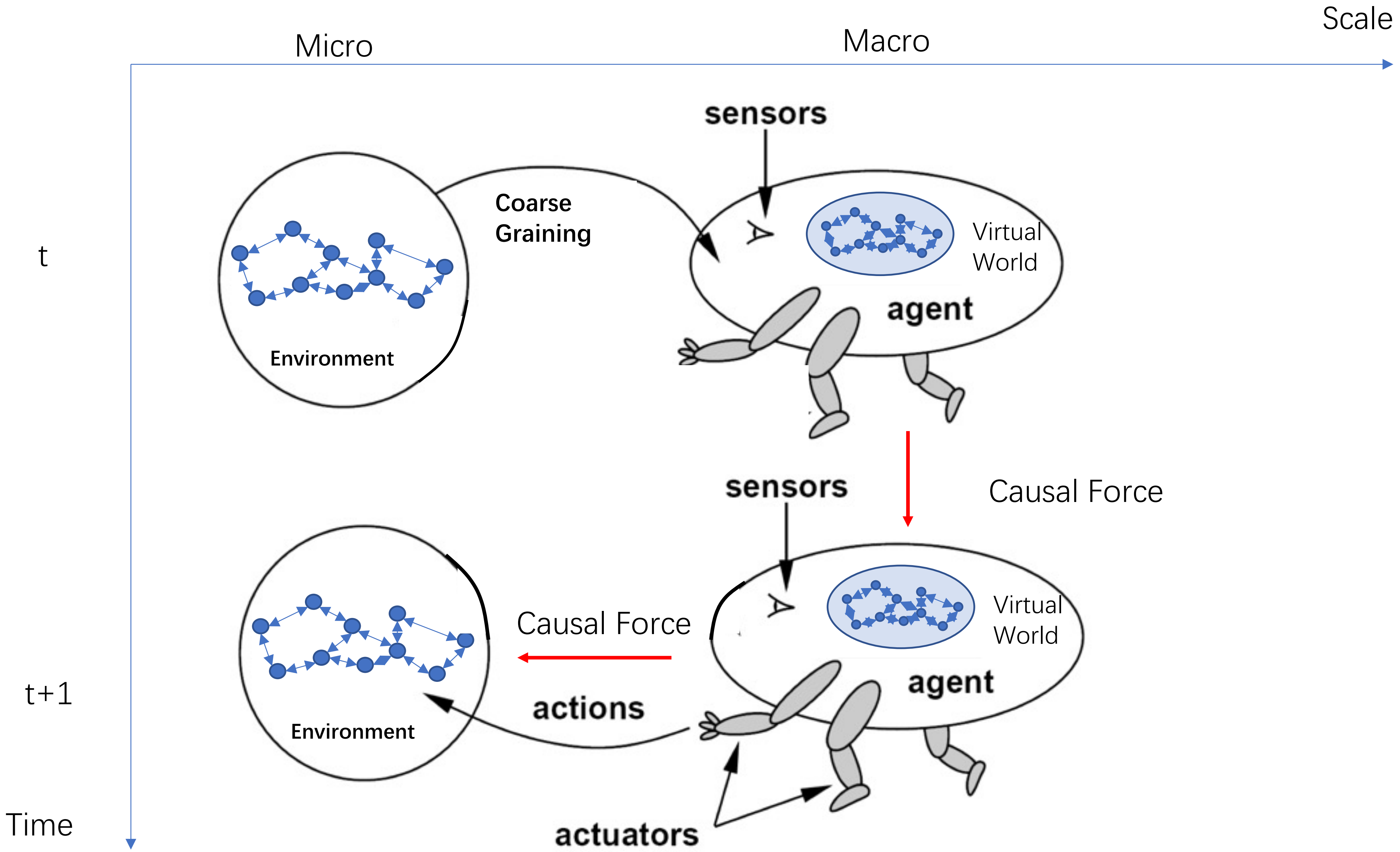
Inner Simulation



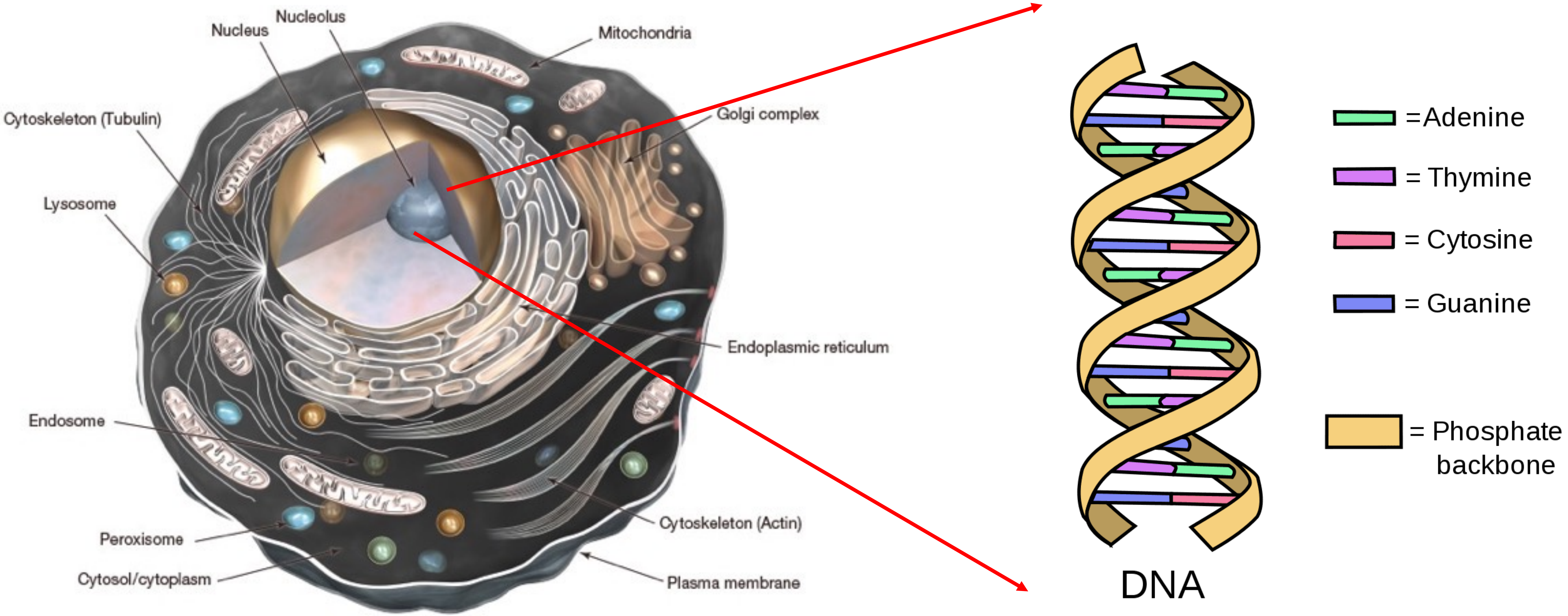
Cognitive and Effective System



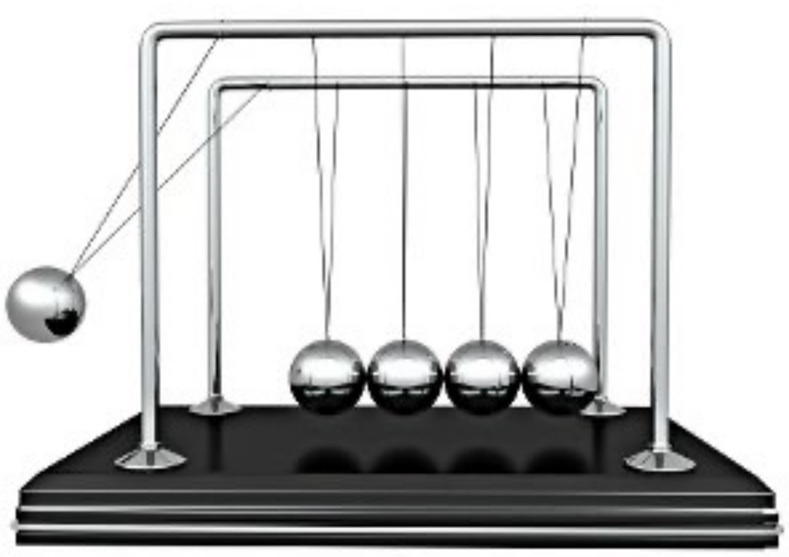
Inner Simulation

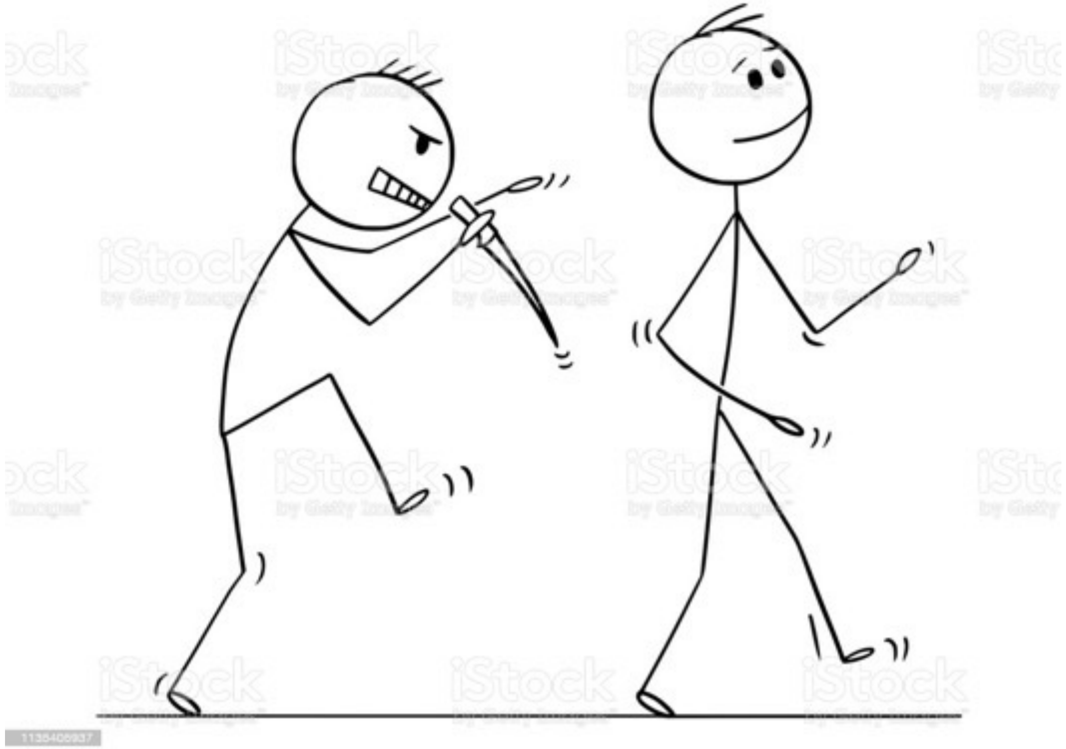


It is general

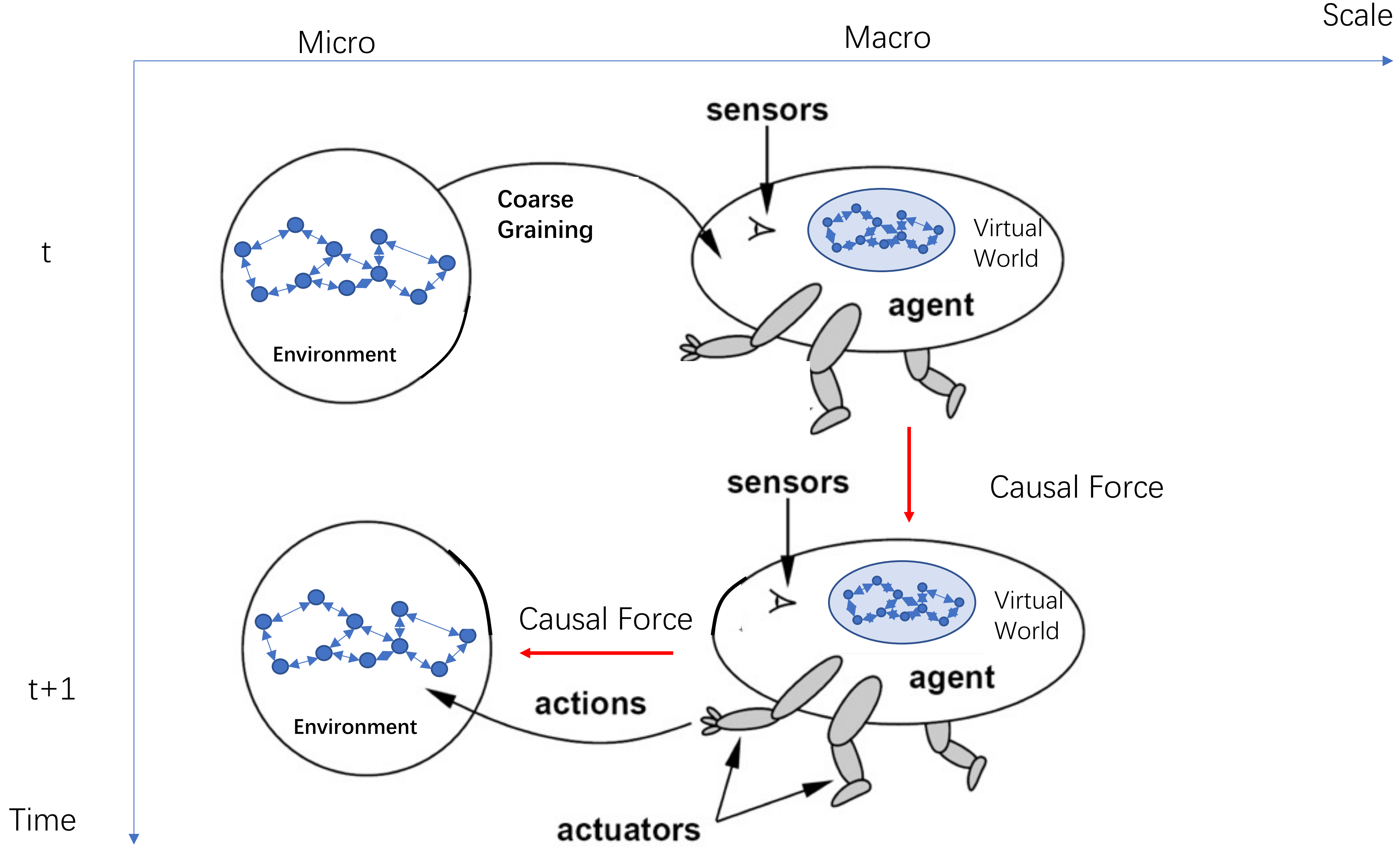


Two Causal Worlds

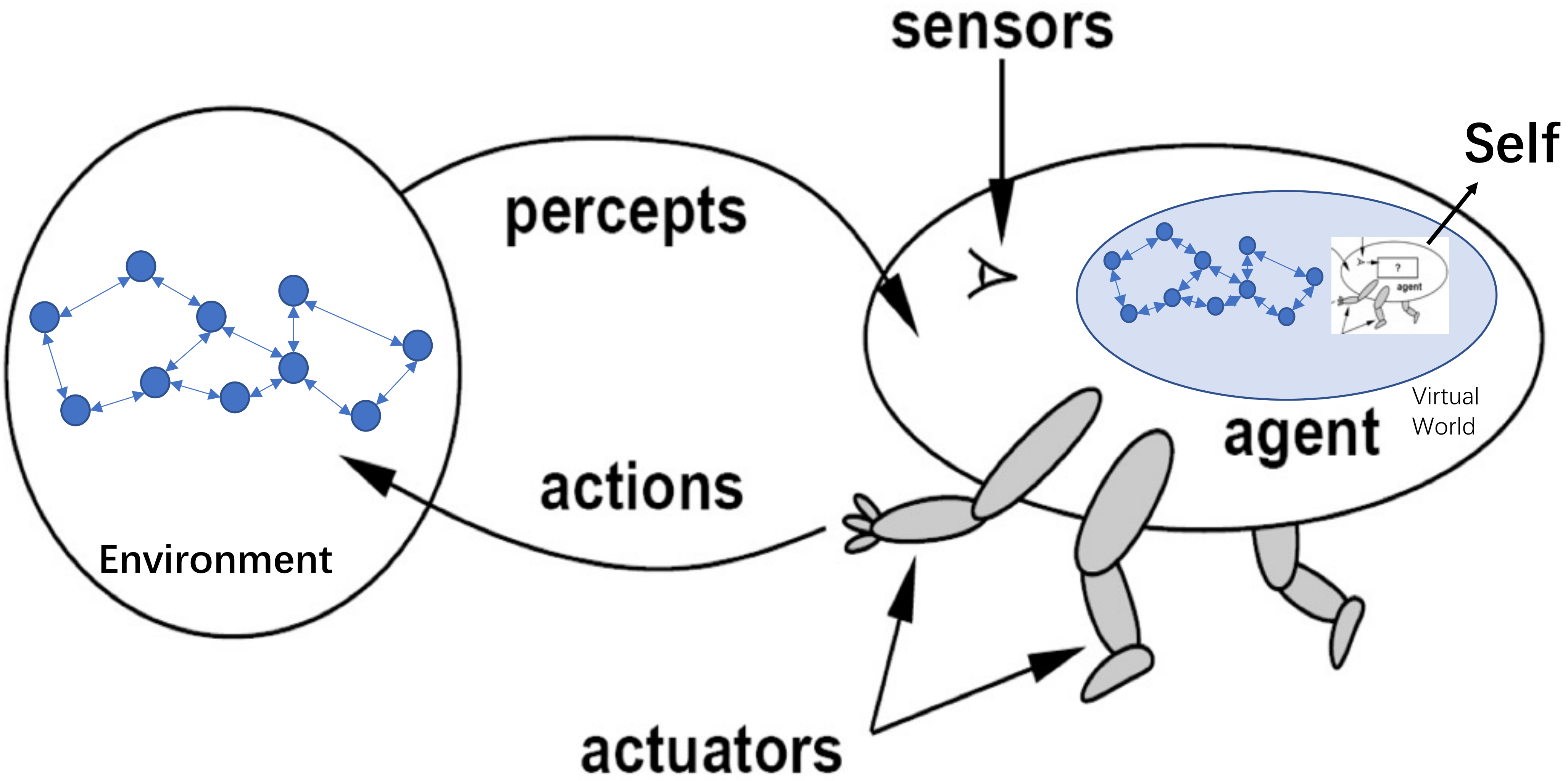




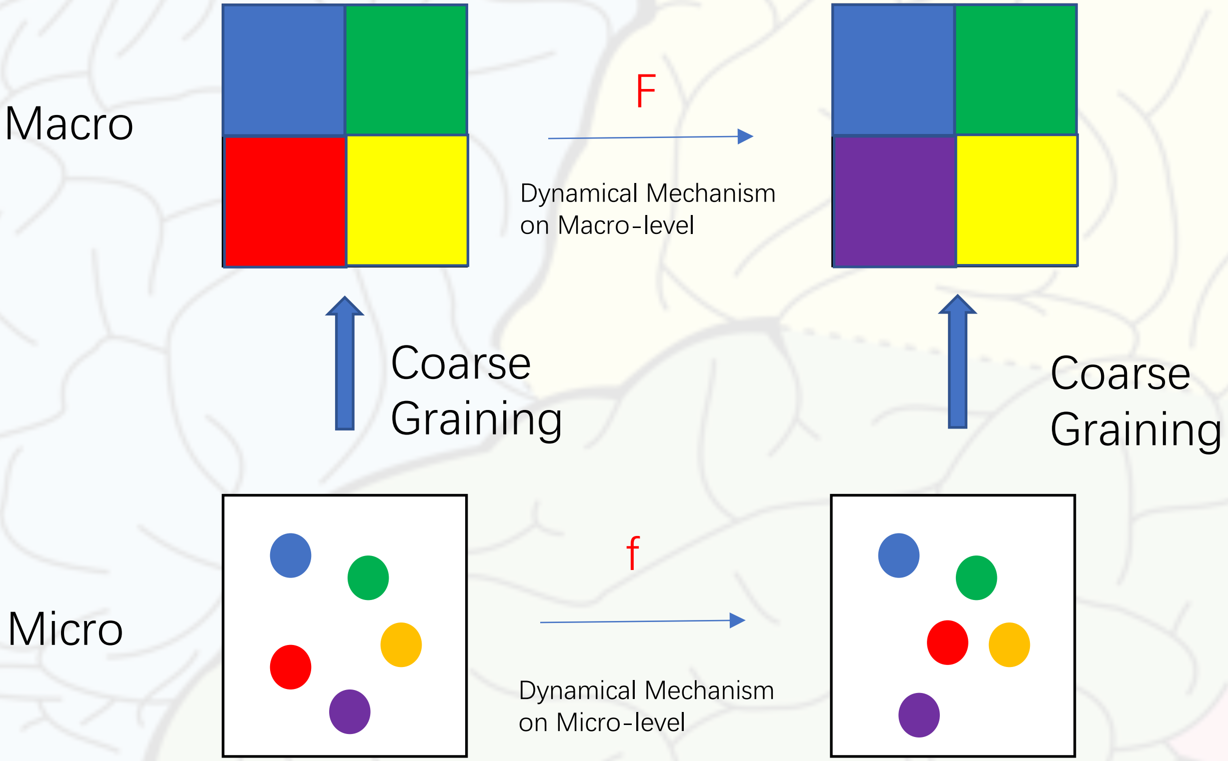
Free Will?


Scale
Micro
Macro
t
t+1
Time
sensors
Coarse Graining
Environment
Virtual World
agent
Causal Force
sensors
Virtual World
agent
Causal Force
actions
Environment
actuators

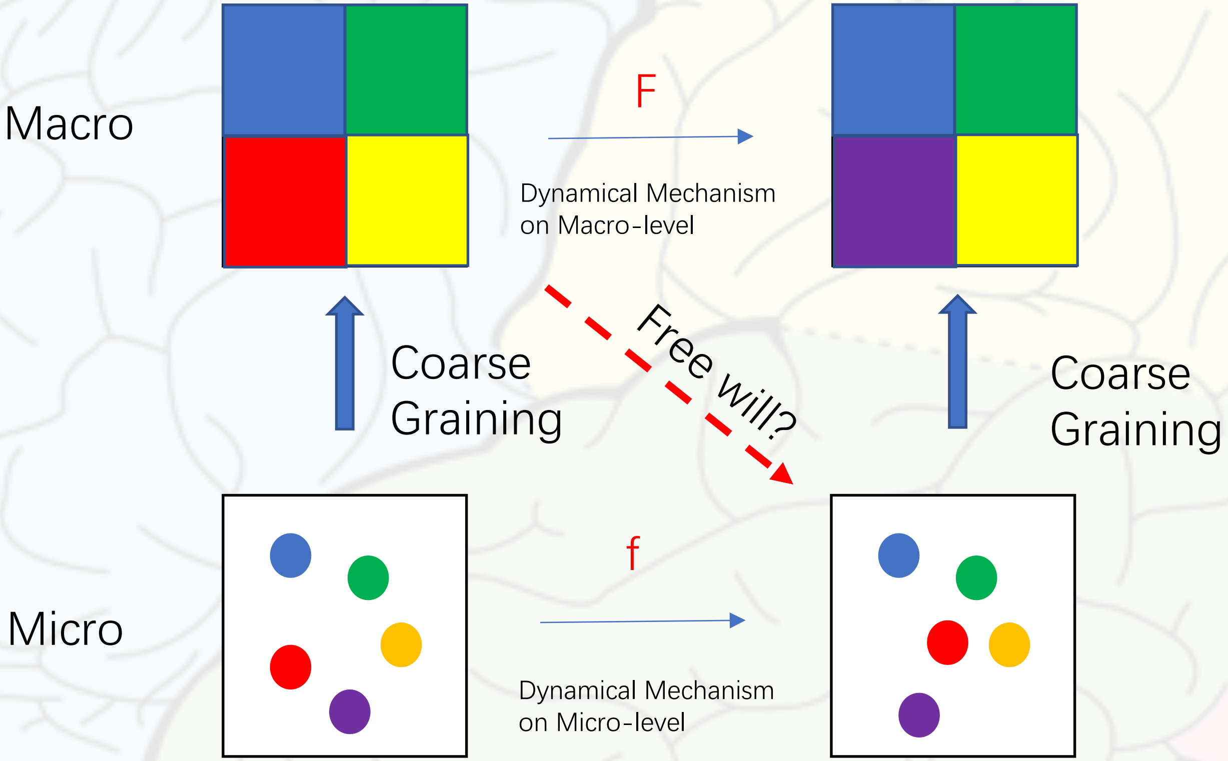
Cognitive System



Self-reference


Macro
F
Dynamical Mechanism on Macro-level
Coarse Graining
Coarse Graining
Micro
f
Dynamical Mechanism on Micro-level

Inner Causal Emergence

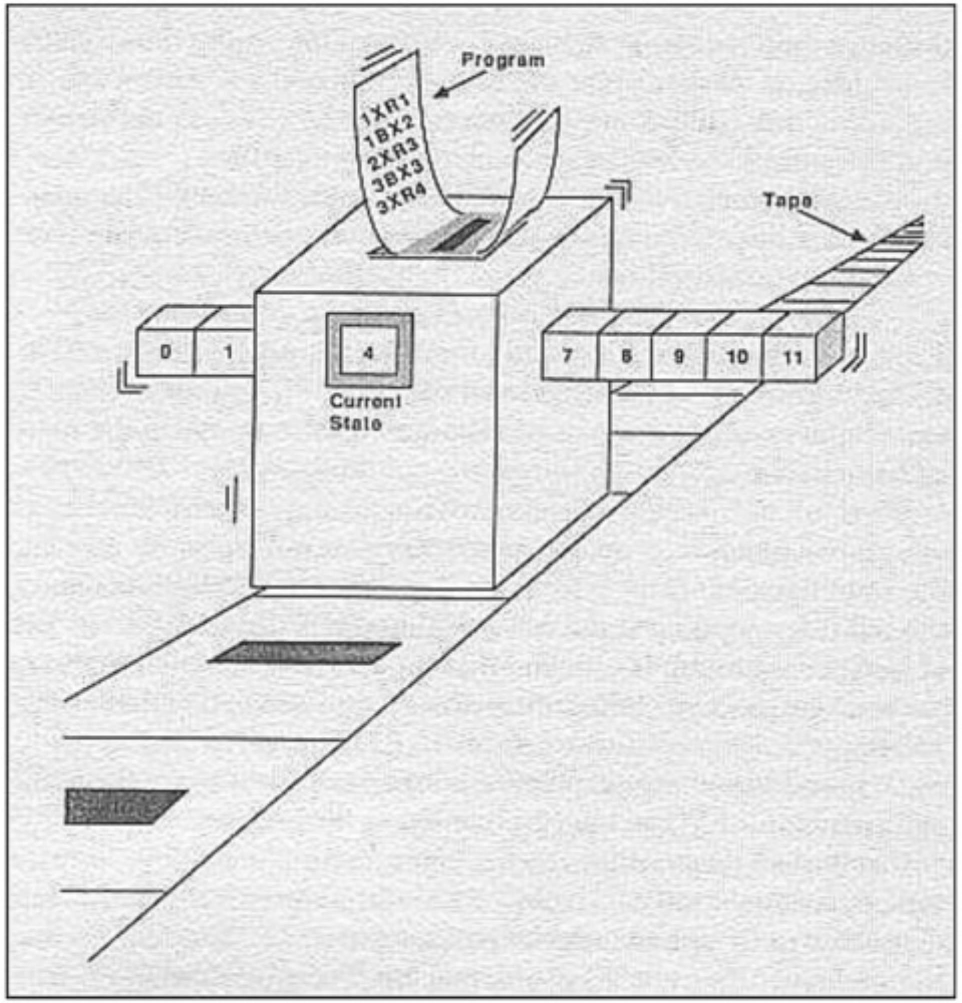

Macro
F
Dynamical Mechanism on Macro-level
Coarse Graining
Free will?
Coarse Graining
Micro
f
Dynamical Mechanism on Micro-level

Inner Causal Emergence

Outline

- Why Self-reference?
- Self-reference in Computational Theory
- Self-referential Dynamics

Turing Machine



- Turing machine can be described as a tuple

- Q: set of states
- A: An alphabet on the tape
- b: The blank symbol on the tape
- $X = A/b$, all input symbols
- $I: X \times (Q/q_F) \rightarrow X \times Q \times \{L, R\}$
- S: output actions
- q_0: the initial state
- q_F: the halt state

Computation by TM

t0

2	2	1	0	1	2	2

T

1	2	2	0	0	1	1

$$Y = f_{TM}(X)$$

Computable functions

- Consider f as a partial function from $\mathcal{N}^n$ to $\mathcal{N}$
- The function f is **TM-computable** if there is a Turing Machine that TM-computes f.
- What is TM-computable?
  - Suppose P is a TM, and let $a_1, a_2, a_3, \ldots, a_n, b \in \mathcal{N}$
    - The computation $P(a_1, a_2, a_3, \ldots, a_n)$ converges to b if $P(a_1, a_2, a_3, \ldots, a_n) \downarrow$ and in the final configuration b is on the tape, then we write $P(a_1, a_2, a_3, \ldots, a_n) \downarrow b$
    - P TM-computes f if, for every $a_1, a_2, a_3, \ldots, a_n, b$, $P(a_1, a_2, a_3, \ldots, a_n) \downarrow b$ if and only if $(a_1, a_2, a_3, \ldots, a_n) \in Dom(f)$ and $f(a_1, a_2, a_3, \ldots, a_n) = b$

Quiz

- Is there a one-one map between TM-computable function and TM?

Effective Denumerable

- Let X be a set of objects, then X is effectively denumerable if there is a bijection $f: X \rightarrow \mathcal{N}$ such that both f and f^{-1} are TM-computable functions.
- The set of all English words is effectively denumerable
  - About←→ 0102152120
  - …

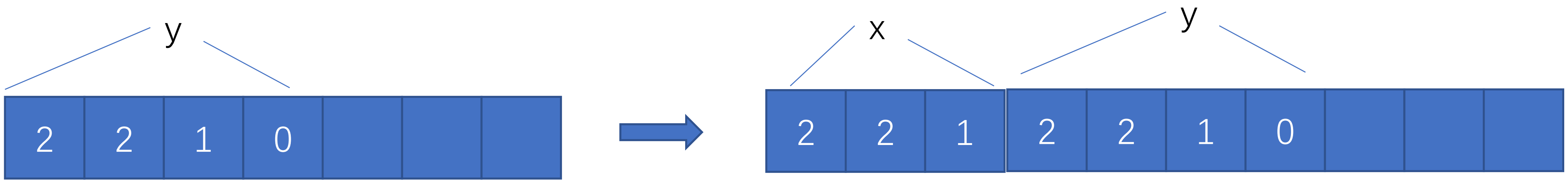
TMs are effective denumerable

- The set of all Turing Machines are effective denumerable
    - Programming is enumeration!

- Define: $\phi_a(x)$ as the corresponding TM-computable function of the Turing Machine enumerated by a.

The s-m-n theorem

- Suppose that $f(x, y)$ is a computable function. There is a total computable function $k(x)$ such that:
  - $f(x, y) \simeq \phi_{k(x)}(y)$
  - $\simeq$ means for given x and y, $f(x, y) = \phi_{k(x)}(y)$
- Proof
  - For each x, k(x) will be the code number of a program Qx which, given initial configuration: y,⋯.
  - Let F be a program that computes f. Then for Qx, we write down F prefaced by instructions that transform the configuration to x,y,⋯.



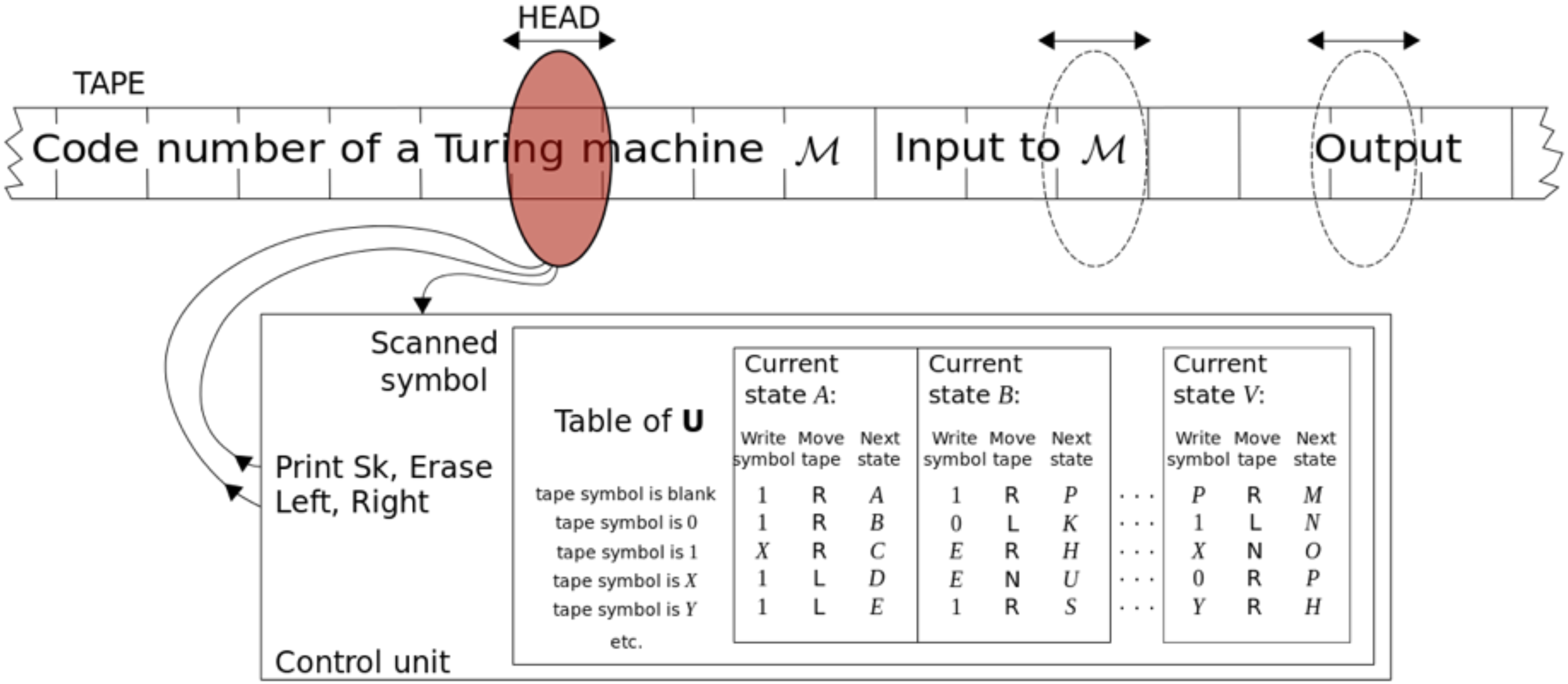
The s-m-n theorem

- S-m-n theorem(simple form)
  - Suppose that $f(x, y)$ is a computable function. There is a total computable function $k(x)$ such that:
  - $f(x, y) \simeq \phi_{k(x)}(y)$
  - $\simeq$ means for given x and y, $f(x, y) = \phi_{k(x)}(y)$
- S-m-n theorem(full form):
  - for each m,n >=1 there is a total computable (m+1)-ary function $s_n^m(e, x)$ such that
  - $\phi_e^{(m+n)}(x, y) = \phi_{s_n^m(e,x)}(y)$

Universal functions

- Definition of universal function:
  - for n-ary computable functions, the universal function is the (n+1)-ary function $\psi_U(e, x_1, x_2, \dots, x_n) = \phi_e(x_1, x_2, \dots, x_n)$
- Theorem:
  - For each n, the universal function ψ_U is computable
- Any program to compute ψ_U is universal program

Universal Turing Machine



	Current state A:			Current state B:				Current state V:		
	Write symbol	Move tape	Next state	Write symbol	Move tape	Next state		Write symbol	Move tape	Next state
tape symbol is blank	1	R	A	1	R	P	. . .	P	R	M
tape symbol is 0	1	R	B	0	L	K	. . .	1	L	N
tape symbol is 1	X	R	C	E	R	H	. . .	X	N	O
tape symbol is X	1	L	D	E	N	U	. . .	0	R	P
tape symbol is Y	1	L	E	1	R	S	. . .	Y	R	H
etc.										

Halting Problem

- For any TM x and input data y, the problem "$\phi_x(y)$ is defined" (equivalently, $P_x(y)\downarrow$) is undecidable
  - Undecidable means:
  - if we define:
  - $g(x,y)=\begin{cases}1 & if\,\phi_x(y)\ is\ defined\\ 0 & if\ \phi_x(y)\ is\ not\ defined\end{cases}$
  - Then, g(x,y) is not computable

Diagonalization method

- Suppose $g(x, y)$ is computable, then we can enumerate all pairs of x and y, and we can get a table

$g(x, y)$

X\Y	1	2	3	4	…
1	1	0	0	1	…
2	0	1	1	0	…
3	1	1	0	0	…
4	0	1	1	0	…
…	…	…	…	…	…

We then can define a new function:

$h(x) = 1 - g(x, x)$

$h(x)'s$ coding is h, then h must be in the table, however, h cannot be in the table because they are not identical to the diagonalization

Another proof

- Suppose P exists
- Then we can construct a program Q which will invoke P

```
Program Q(X){
        m=P(X,X)
        do while (m=False)
                …
                …
        end do
        if m=True then return
```

- What will happen if Q(Q)

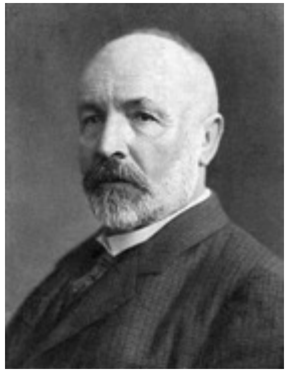
Cantor's proof

- The set of real number is not countable

Suppose we can enumerate real numbers in [0,1]

- 0.38483905784923928
- 0.54939843944343434
- 0.88434889540543829
- 0.32904304394394993
- 0.54954923892895456
- 0.43968598938933232
- …

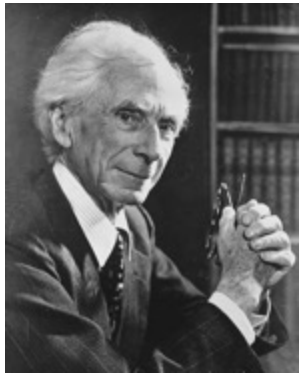
- T=0.455156⋯



Cantor

Russel’s paradox

- $S = \{A | A \notin A\}$
- How about $S \in S$?
    - If $S \in S$, then according to its own rule, $S \notin S$
    - But, if $S \notin S$, then $S \in S$ according to its own rule



Russel

Godel’s second incompleteness theorem

- Second incompleteness theorem:
  - Assume F is consistent formalized system which contains elementary arithmetic, then: $F \nvdash \mathbf{Cons}(F)$
  - If the system is consistent, it cannot be complete.
  - The consistency of the axioms cannot be proven within the system.
- Godel’s proof
  - This sentence is un-provable in F

Kurt Godel

1906-1978

Kleene’s second recursion theorem

- Let f be a total unary computable function; then there is a number n such that

$$\phi_{f(n)} = \phi_n$$

- Any number of n such that $\phi_{f(n)} = \phi_n$ called fixed point

Proof

- According to the s-m-n theorem, there is a total computable function s(x), such that for all x
- $\phi_{f(\phi_x(x))}(y) = \phi_{s(x)}(y)$
    - $\phi_{f(\phi_z(x))}(y) \equiv v(x, y) \simeq \phi_{s(x)}(y)$
- Because $s(x)$ is computable, let m be the number of this program,
- $\phi_{f(\phi_x(x))}(y) = \phi_{\phi_m(x)}(y)$
- Now, let x=m, then:
- $\phi_{f(\phi_m(m))}(y) = \phi_{\phi_m(m)}(y)$
- So, let $n = \phi_m(m)$, then
- $\phi_{f(n)} = \phi_n$

Self-print program

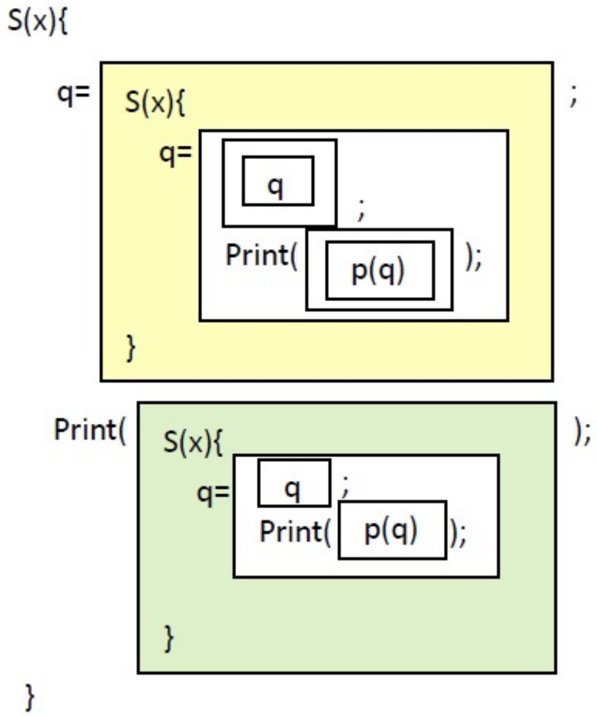
- Let f be printing program
  - f(x): print x on the tape
- Then, there exists a number n, such that $\phi_{f(n)} = \phi_n$
  - n is the code(source code) of the program P
  - P defines a TM -computable function ϕ_n
  - f(n): print n(source code of P) p on tape
  - $\phi_{f(n)} = \phi_n$ means P can print itself source code

Quine

- There is self-print program

Quine

```
S (x){
  q=’S(x){\\n  q=\\\’\’+q+\’\\\’;\\n  Print(\\\’\’+p(q)+\’\\\’);\\n}’;
  Print (‘S(x){\n  q=\”+q+’ \’;\n  Print(\”+p(q)+’\’);\n}’);
}
```



Diagonalization method

Data

TMs

X\Y	1	2	3	4	⋯
1	$\phi_{\phi_1(1)}$	$\phi_{\phi_1(2)}$	$\phi_{\phi_1(3)}$	$\phi_{\phi_1(4)}$	⋯
2	$\phi_{\phi_2(1)}$	$\phi_{\phi_2(2)}$	$\phi_{\phi_2(3)}$	$\phi_{\phi_2(4)}$	⋯
3	$\phi_{\phi_3(1)}$	$\phi_{\phi_3(2)}$	$\phi_{\phi_3(3)}$	$\phi_{\phi_3(4)}$	⋯
4	$\phi_{\phi_4(1)}$	$\phi_{\phi_4(2)}$	$\phi_{\phi_4(3)}$	$\phi_{\phi_4(4)}$	⋯
⋯	⋯	⋯	⋯	⋯	⋯

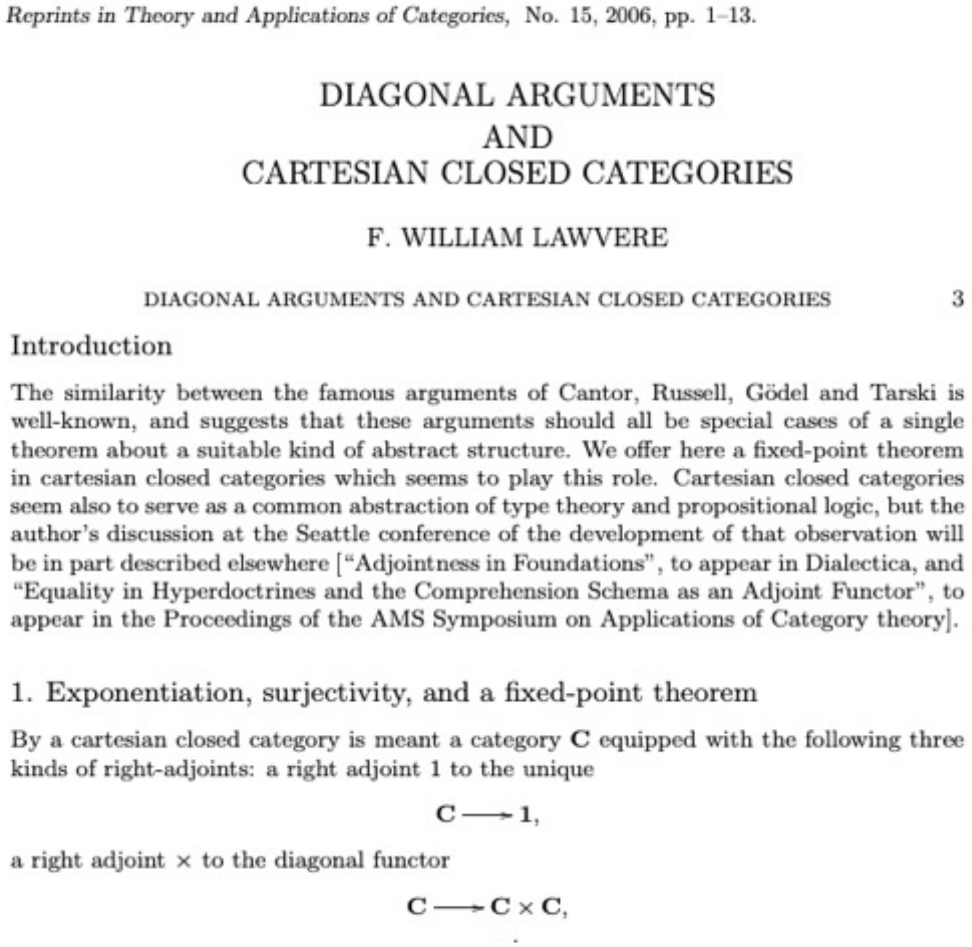
Diagonalization

Data

TMs

X\Y	1	2	3	4	⋯	m	⋯
1	$\phi_{\phi_1(1)}$	$\phi_{\phi_1(2)}$	$\phi_{\phi_1(3)}$	$\phi_{\phi_1(4)}$	⋯	$\phi_{\phi_1(m)}$	⋯
2	$\phi_{\phi_2(1)}$	$\phi_{\phi_2(2)}$	$\phi_{\phi_2(3)}$	$\phi_{\phi_2(4)}$	⋯	$\phi_{\phi_2(m)}$	⋯
3	$\phi_{\phi_3(1)}$	$\phi_{\phi_3(2)}$	$\phi_{\phi_3(3)}$	$\phi_{\phi_3(4)}$	⋯	$\phi_{\phi_3(m)}$	⋯
4	$\phi_{\phi_4(1)}$	$\phi_{\phi_4(2)}$	$\phi_{\phi_4(3)}$	$\phi_{\phi_4(4)}$	⋯	$\phi_{\phi_4(m)}$	⋯
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
m	$\boldsymbol{\phi_{f(\phi_1(1))}}$	$\boldsymbol{\phi_{f(\phi_2(2))}}$	$\boldsymbol{\phi_{f(\phi_3(3))}}$	$\boldsymbol{\phi_{f(\phi_4(4))}}$	**⋯**	$\boldsymbol{\phi_{\phi_m(m)} = \phi_{f(\phi_m(m))}}$	**⋯**
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯

For any f, $f(\phi_x(x))$ must be in one row because f, $\phi_x(x)$ are computable



Reprints in Theory and Applications of Categories, No. 15, 2006, pp. 1–13.

DIAGONAL ARGUMENTS
AND
CARTESIAN CLOSED CATEGORIES

F. WILLIAM LAWVERE

Introduction

The similarity between the famous arguments of Cantor, Russell, Gödel and Tarski is well-known, and suggests that these arguments should all be special cases of a single theorem about a suitable kind of abstract structure. We offer here a fixed-point theorem in cartesian closed categories which seems to play this role. Cartesian closed categories seem also to serve as a common abstraction of type theory and propositional logic, but the author's discussion at the Seattle conference of the development of that observation will be in part described elsewhere ["Adjointness in Foundations", to appear in Dialectica, and "Equality in Hyperdoctrines and the Comprehension Schema as an Adjoint Functor", to appear in the Proceedings of the AMS Symposium on Applications of Category theory].

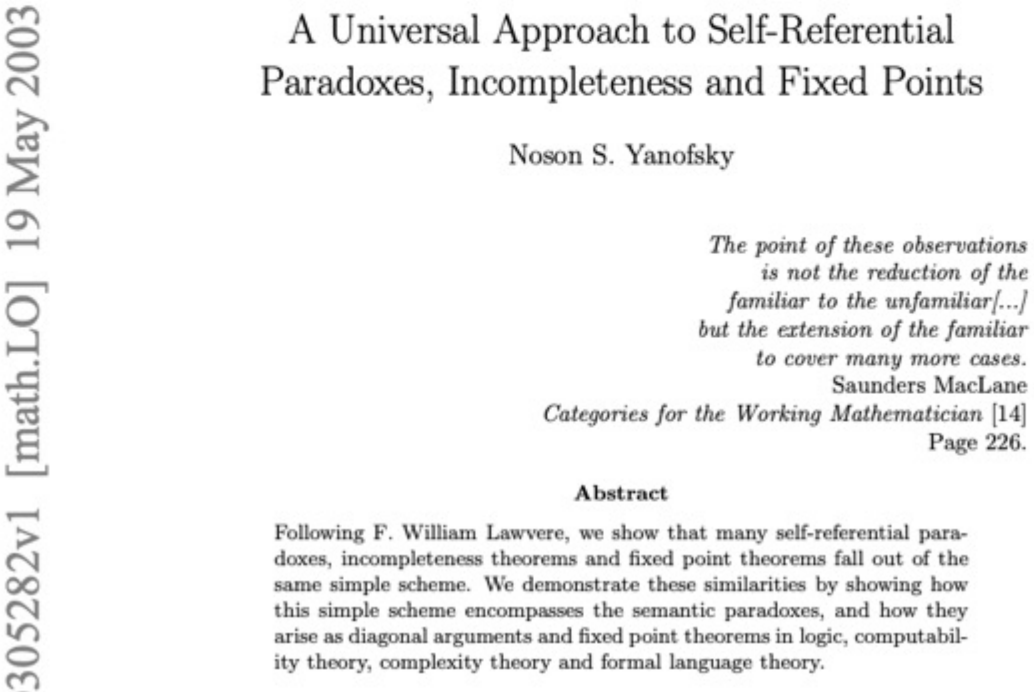
1. Exponentiation, surjectivity, and a fixed-point theorem

By a cartesian closed category is meant a category $\mathbf{C}$ equipped with the following three kinds of right-adjoints: a right adjoint 1 to the unique

$$\mathbf{C} \longrightarrow \mathbf{1},$$

a right adjoint × to the diagonal functor

$$\mathbf{C} \longrightarrow \mathbf{C} \times \mathbf{C},$$



A Universal Approach to Self-Referential Paradoxes, Incompleteness and Fixed Points

Noson S. Yanofsky

> *The point of these observations*
> *is not the reduction of the*
> *familiar to the unfamiliar[...]*
> *but the extension of the familiar*
> *to cover many more cases.*
> Saunders MacLane
> *Categories for the Working Mathematician* [14]
> Page 226.

Abstract

Following F. William Lawvere, we show that many self-referential paradoxes, incompleteness theorems and fixed point theorems fall out of the same simple scheme. We demonstrate these similarities by showing how this simple scheme encompasses the semantic paradoxes, and how they arise as diagonal arguments and fixed point theorems in logic, computability theory, complexity theory and formal language theory.

Lawvere(cantor) theorem

- Theorem: If Y is a set and there exists a function $\alpha\colon T \to T$ without a fixed point (for all $t \in T$, $\alpha(t) \neq t$), then for all sets Y and for all functions $f\colon Y \times Y \to T$, there exists a function $g\colon Y \to T$ that is not representable by f, i.e., such that for all $y \in Y$, $g(\cdot) \neq f(y,\ \cdot)$

Lawvere(cantor) theorem

- Theorem: If Y is a set and there exists a function $\alpha\colon T \to T$ without a fixed point (for all $t \in T$, $\alpha(t) \neq t$), then for all sets Y and for all functions $f\colon Y\times Y \to T$, there exists a function $g\colon Y \to T$ that is not representable by f, i.e., such that for all $y \in Y$, $g(\cdot) \neq f(y,\cdot)$

$$\begin{tikzcd}
Y\times Y \arrow[r, "f"] & T \arrow[d, "\alpha"] \\
Y \arrow[u, "\Delta"] \arrow[r, "g"'] & T
\end{tikzcd}$$

Proof:

- There is a function $\Delta\colon Y \to Y\times Y$ that sends every y to $(y, y) \in Y\times Y$.
- Then construct $g\colon Y \to T$ as the composition of the three functions $g(y) = \alpha(f(y, y))$,
- We claim that for all $y \in Y$, $g(\cdot) \neq f(y,\cdot)$ as function of one variable.
- If $g(\cdot) = f(y_0,\ \cdot)$ then by evaluation at y_0, we have: $f(y_0, y_0) = g(y_0) = \alpha(f(y_0, y_0))$, which is the fixed point of α, contradicts!

TM halting problem

- Theorem: If Y is a set and there exists a function $\alpha\colon T \to T$ without a fixed point (for all $t \in T$, $\alpha(t) \neq t$), then for all sets Y and for all functions $f\colon Y{\times}Y \to T$, there exists a function $g\colon Y \to T$ that is not representable by f, i.e., such that for all $y \in Y$, $g(\cdot) \neq f(y,\ \cdot)$

$$\begin{tikzcd}
Y{\times}Y \arrow[r, "f"] & T \arrow[d, "\alpha"] \\
Y \arrow[u, "\Delta"] \arrow[r, "g"'] & T
\end{tikzcd}$$

(TM: m, input: n)

$$\begin{tikzcd}
N{\times}N \arrow[r, "Halt"] & \{0,1\} \arrow[d, "\sim(Not)"] \\
N \arrow[u, "\Delta"] \arrow[r, "g"'] & \{0,1\}
\end{tikzcd}$$

$$Halt(m,n) = \begin{cases} 1 & if\,\phi_m(n)\ is\ defined \\ 0 & if\ not \end{cases}$$

$1 - Halt(m,n)$

Codes of TMs

TM halting problem

$$\begin{tikzcd}
Y \times Y \arrow[r, "f"] & T \arrow[d, "\alpha"] \\
Y \arrow[u, "\Delta"] \arrow[r, "g"] & T
\end{tikzcd}$$

(TM: m, input: n)

$$\begin{tikzcd}
N \times N \arrow[r, "Halt"] & \{0,1\} \arrow[d, "\alpha"] \\
N \arrow[u, "\Delta"] \arrow[r, "g"] & \{\uparrow, 1\}
\end{tikzcd}$$

Codes of TMs

$$Halt(m,n) = \begin{cases} 1 & if\, \phi_m(n)\ is\ defined \\ 0 & if\ not \end{cases}$$

$$\alpha(x) = \begin{cases} 1 & x = 0 \\ \uparrow & x = 1 \end{cases}$$

If there exists y_0 such that: $g(\cdot) = f(y_0,\cdot)$

If there exists y_0 such that $g(\cdot) = Halt(y_0,\cdot)$

y_0 is the coding of g

$$f(y_0, y_0) = g(y_0) = \alpha(f(y_0, y_0))$$

$$Halt(y_0, y_0) = g(y_0) = \alpha(Halt(y_0, y_0))$$

Russel Paradox

$$\begin{tikzcd}
Y\times Y \arrow[r, "f"] & T \arrow[d, "\alpha"] \\
Y \arrow[u, "\Delta"] \arrow[r, "g"] & T
\end{tikzcd}$$

If there exists y_0 such that: $g(\cdot) = f(y_0,\cdot)$

$$f(y_0, y_0) = g(y_0) = \alpha(f(y_0, y_0))$$

S as set, S as element

$$\begin{tikzcd}
 & S\times S \arrow[r, "x \in X"] & \{0,1\} \arrow[d, "\sim(Not)"] & \\
\text{All sets} & S \arrow[u, "\Delta"] \arrow[r, "g"] & \{0,1\} & \sim\chi(X,x)
\end{tikzcd}$$

$$\chi(X, x) = \begin{cases} 1 & if\ x \in X \\ 0 & if\ not \end{cases}$$

If there exists y_0 such that: $g(\cdot) = \chi(y_0,\cdot)$

y_0 is the set that
not contains itself

$$\chi(y_0, y_0) = g(y_0) = \sim(\chi(y_0, y_0))$$

Diagonal Theorem

- Theorem: If T is a set and there exists a set Y and a function $f\colon Y \times Y \to T$ such that all functions $g\colon Y \to T$ are representable by f, i.e., there exists a $y \in Y$, such that $g(\cdot) = f(y, \cdot)$, then for all functions $\alpha\colon T \to T$, there is a fixed point

Diagonal Theorem

- Theorem: If T is a set and there exists a set Y and a function $f\colon Y\times Y \to T$ such that all functions $g\colon Y \to T$ are representable by f, i.e., there exists a $y \in Y$, such that $g(\cdot) = f(y,\cdot)$, then for all functions $\alpha\colon T \to T$, there is a fixed point

$$\begin{tikzcd}
Y\times Y \arrow[r, "f"] & T \arrow[d, "\alpha"] \\
Y \arrow[u, "\Delta"] \arrow[r, "g"] & T
\end{tikzcd}$$

Proof:

We construct g as follows:

$$g(m) = \alpha(f(m,m))$$

Since we have assumed that g is representable by some $y \in Y$, we have that

$$g(m) = f(y,m)$$

So, we have a fixed point of α at $t_0 = g(y)$, explicitly:

$$\alpha\big(g(y)\big) = \alpha\big(f(y,y)\big) = g(y)$$

Recursion Theorem(my version)

$$\begin{tikzcd}
Y\times Y \arrow[r, "f"] & T \arrow[d, "\alpha"] \\
Y \arrow[u, "\Delta"] \arrow[r, "g"] & T
\end{tikzcd}$$

$$\begin{tikzcd}
\text{(TM: m, input: n)} & N\times N \arrow[r, "f(m,n)"] & N \arrow[d, "h"] & \text{Coding all functions on } \phi_m(n) \\
\text{Codes of TMs} & N \arrow[u, "\Delta"] \arrow[r, "g(m) \equiv h(f(m,m))"'] & N &
\end{tikzcd}$$

Recursion Theorem(my version)

$$\begin{tikzcd}
Y\times Y \arrow[r, "f"] & T \arrow[d, "\alpha"] \\
Y \arrow[u, "\Delta"] \arrow[r, "g"] & T
\end{tikzcd}$$

(TM: m, input: n)

$$\begin{tikzcd}
N\times N \arrow[r, "f(m,n)"] & N \arrow[d, "h"] \\
N \arrow[u, "\Delta"] \arrow[r, "g(m) \equiv h\left(f(m,m)\right)"'] & N
\end{tikzcd}$$

Coding all functions on $\phi_m(n)$

Codes of TMs

we have a fixed point of α at $t_0 = g(y)$, explicitly:

$$\alpha\left(g(y)\right) = \alpha\left(f(y,y)\right)$$
$$= g(y)$$

Fixed point: $t_0 = g(y)$

y is the representation of g under f, $f(y,.) = g(.)$

$$h(t_0) = h\left(g(y)\right) = h(f(y,y)) = g(y)$$

Recursion Theorem(Yanofsky’s version)

$$\begin{tikzcd}
Y \times Y \arrow[r, "f"] & T \arrow[d, "\alpha"] \\
Y \arrow[u, "\Delta"] \arrow[r, "g"] & T
\end{tikzcd}$$

(TM: m, input: n)

All TM-com functions

$$\begin{tikzcd}
N \times N \arrow[r, "f(m,n) = \phi_{\phi_m(n)}"] & \mathrm{F} \arrow[d, "\Phi_h(\phi_n) = \phi_{h(n)}"] \\
N \arrow[u, "\Delta"] \arrow[r, "g(m) \equiv \Phi_h\big(f(m,m)\big) = \phi_{h(\phi_m(m))}"'] & \mathrm{F}
\end{tikzcd}$$

Codes of TMs

Recursion Theorem (Yanofsky’s version)

$$\begin{tikzcd}
Y\times Y \arrow[r, "f"] & T \arrow[d, "\alpha"] \\
Y \arrow[u, "\Delta"] \arrow[r, "g"] & T
\end{tikzcd}$$

(TM: m, input: n)

All TM-com functions

Codes of TMs

$$\begin{tikzcd}
N\times N \arrow[r, "f(m,n) = \phi_{\phi_m(n)}"] & \mathrm{F} \arrow[d, "\Phi_h(\phi_n) = \phi_{h(n)}"] \\
N \arrow[u, "\Delta"] \arrow[r, "g(m) \equiv \Phi_h\left(f(m,m)\right) = \phi_{h(\phi_m(m))}"'] & \mathrm{F}
\end{tikzcd}$$

we have a fixed point of α at $t_0 = g(y)$, explicitly:

$\alpha\left(g(y)\right) = \alpha\left(f(y,y)\right)$

$= g(y)$

Fixed point: $t_0 = g(y)$

y is the representation of g under f,
$f(y,.) = g(.)$

$$\Phi_h(t_0) = \Phi_h\left(g(y)\right) = \Phi_h(f(y,y)) = \Phi_h\left(\phi_{\phi_y(y)}\right) = \phi_{h(\phi_y(y))} = g(y)$$

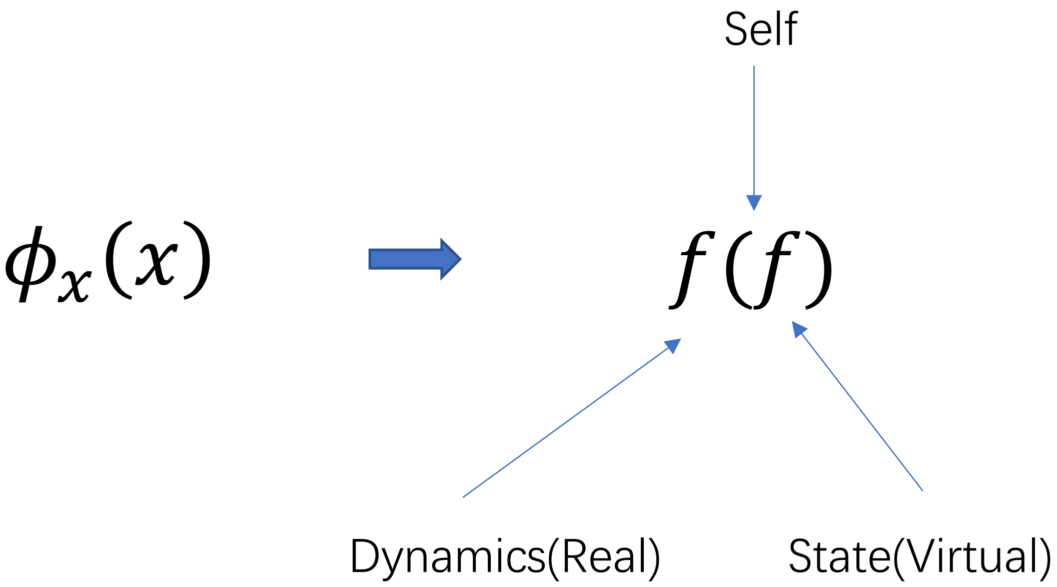
$\phi_{f(n)} = \phi_n \xrightarrow{\qquad}$ Reflection point of causality

ϕ_n n is a proactive program

$\phi_{f(n)}$ n is a passive data

$\phi_{f(n)} = \phi_n$ A passive data n can proactively execute anything

Diagonalization



$\phi_x(x)$ ⇒ $f(f)$

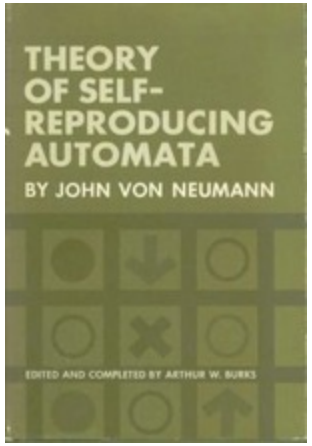
Self

Dynamics(Real)

State(Virtual)

Von Neumann’s Self-reproducing Automata

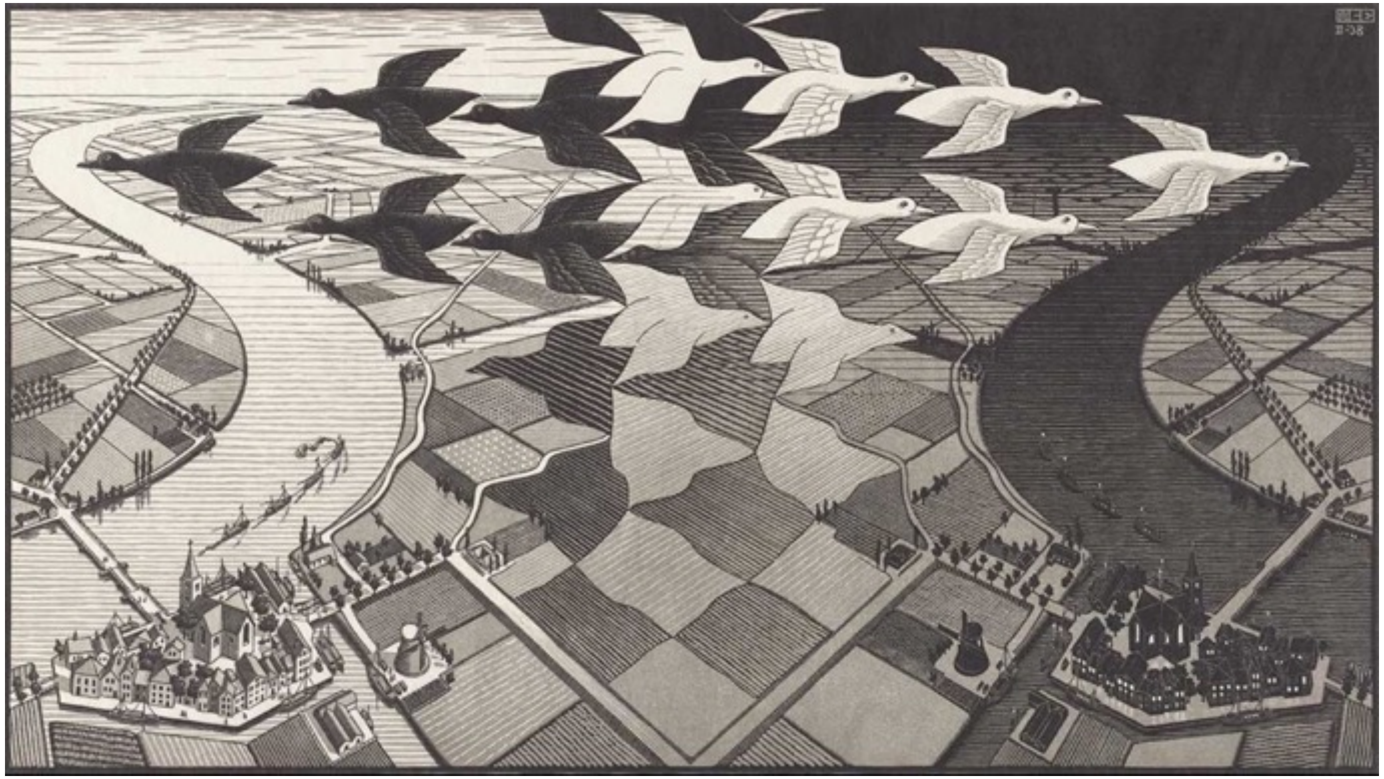
John von Neumann



THEORY OF SELF-REPRODUCING AUTOMATA

BY JOHN VON NEUMANN

EDITED AND COMPLETED BY ARTHUR W. BURKS

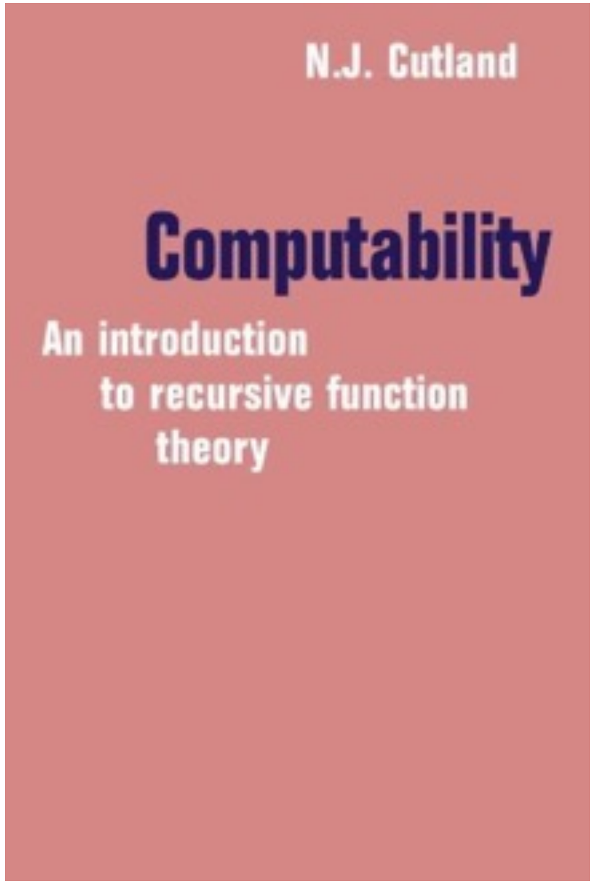


系统的复杂度存在着一个阈值，超过该阈值，系统便可以不断进化下去，达不到则会衰败

Self-retrospection

- 自我意识的核心：自我反省
- 存在着这样一种计算机程序$F_t(x)$，它的作用就是计算任意的源代码为x的程序在经过t时间步的运算后的结果。
- 于是根据递归定理，我们便知道，存在着一个源程序O，它所做的就是：把自己的源代码拿出来，然后在自己的虚拟机上模拟自己运算t时间步后的结果。

Self-awareness = Self-retrospection = Quine + Universal Simulator + IO

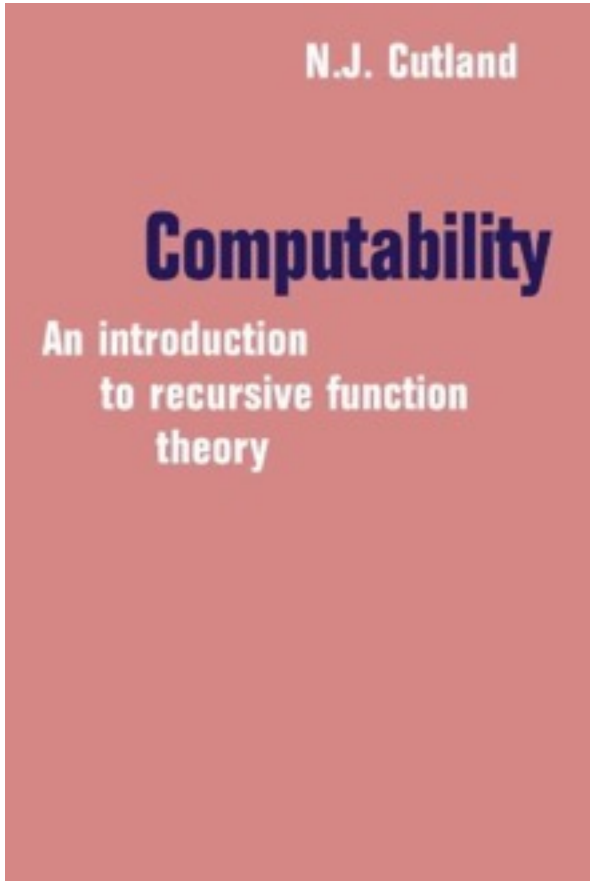


N.J. Cutland

Computability

An introduction to recursive function theory

Reference



N.J. Cutland

Computability

An introduction to recursive function theory

GÖDEL ESCHER BACH

哥德尔 艾舍尔 巴赫

——集异璧之大成

〔美〕侯世达 著

商务印书馆

https://wiki.swarma.org/index.php?title=系统中的观察者(5)——自指

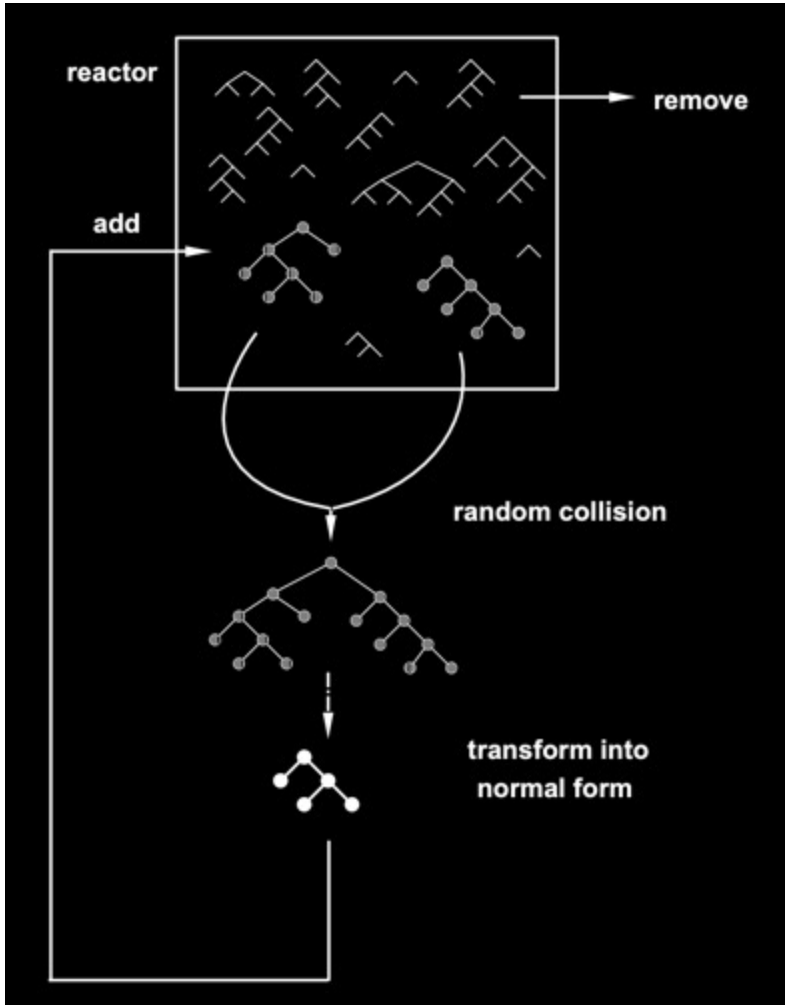
Outline

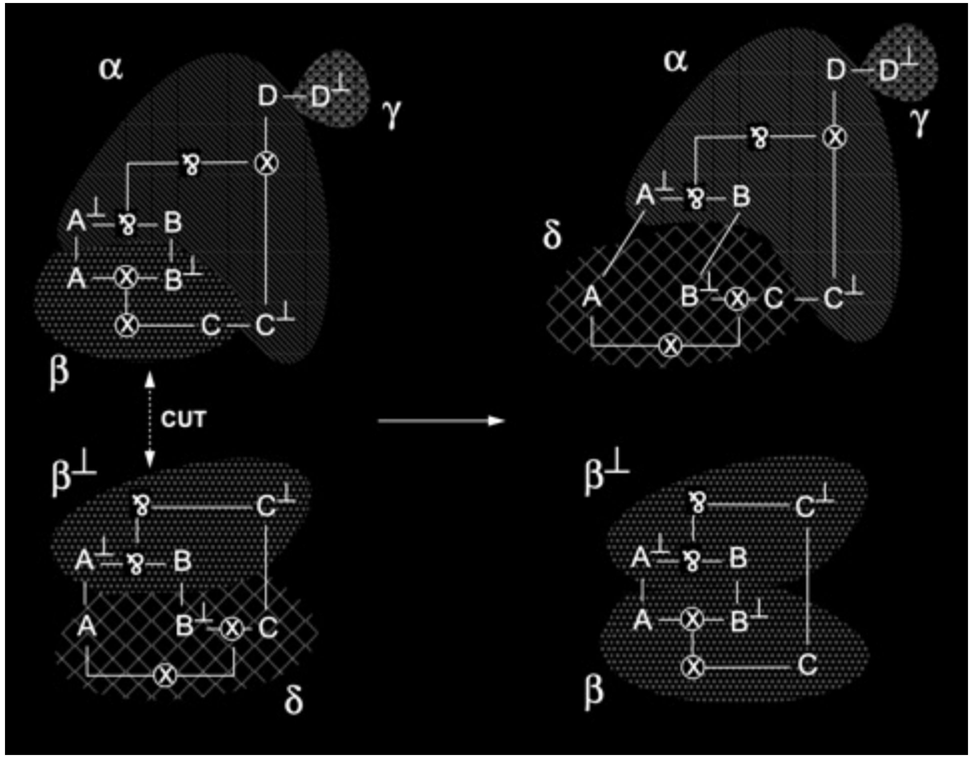
- Why Self-reference?
- Self-reference in Computational Theory
- Self-referential Dynamics

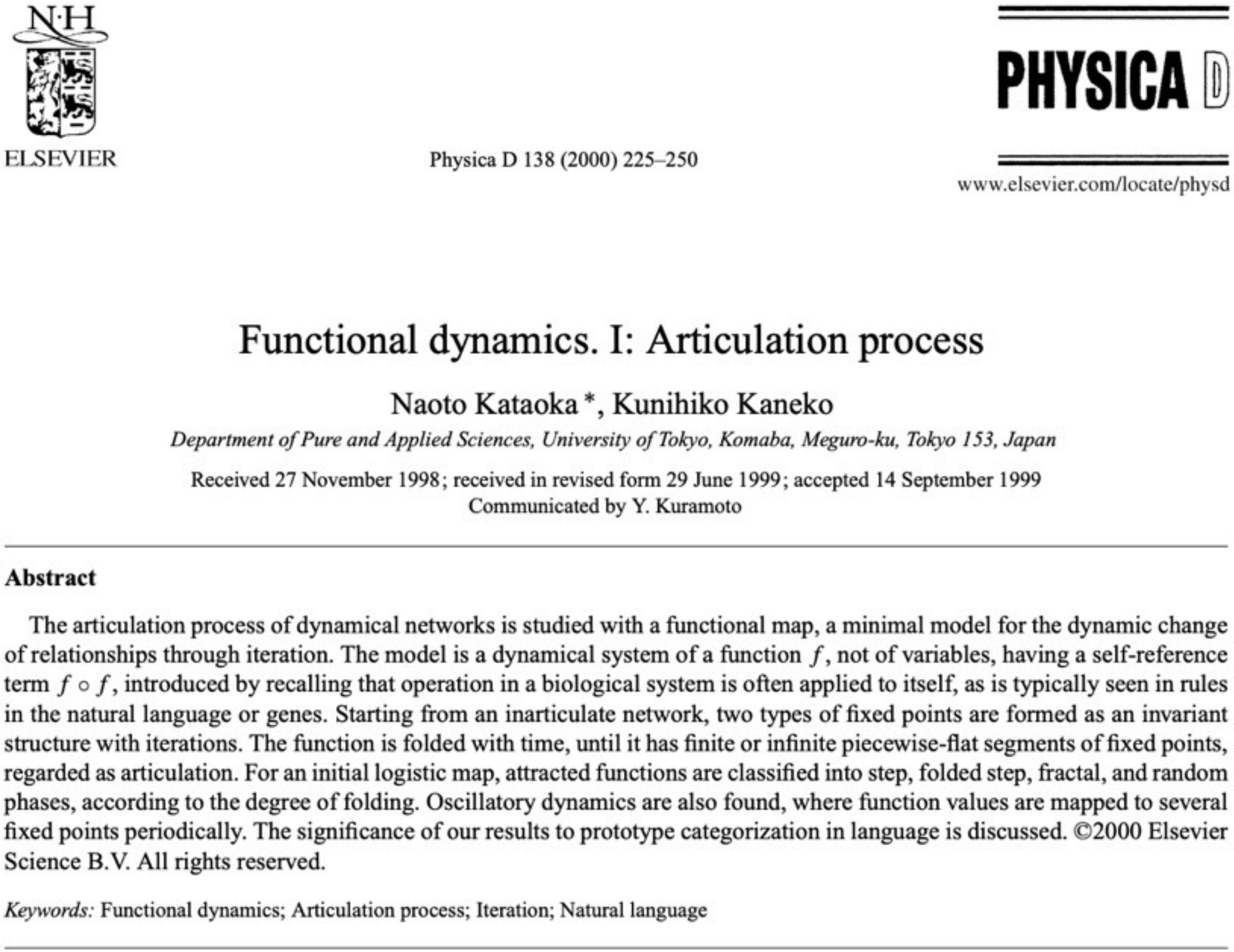
Questions

- Functional dynamics?
- How dynamical system contains self-reference f(f) evolve?
- How self-reference emerge from functional dynamics?

Fontana’s alchemy







ELSEVIER

Physica D 138 (2000) 225–250

PHYSICA D

www.elsevier.com/locate/physd

Functional dynamics. I: Articulation process

Naoto Kataoka *, Kunihiko Kaneko

Department of Pure and Applied Sciences, University of Tokyo, Komaba, Meguro-ku, Tokyo 153, Japan

Received 27 November 1998; received in revised form 29 June 1999; accepted 14 September 1999
Communicated by Y. Kuramoto

Abstract

The articulation process of dynamical networks is studied with a functional map, a minimal model for the dynamic change of relationships through iteration. The model is a dynamical system of a function f, not of variables, having a self-reference term $f \circ f$, introduced by recalling that operation in a biological system is often applied to itself, as is typically seen in rules in the natural language or genes. Starting from an inarticulate network, two types of fixed points are formed as an invariant structure with iterations. The function is folded with time, until it has finite or infinite piecewise-flat segments of fixed points, regarded as articulation. For an initial logistic map, attracted functions are classified into step, folded step, fractal, and random phases, according to the degree of folding. Oscillatory dynamics are also found, where function values are mapped to several fixed points periodically. The significance of our results to prototype categorization in language is discussed.

Keywords: Functional dynamics; Articulation process; Iteration; Natural language

http://chaos.c.u-tokyo.ac.jp/study/papers3.html

Functional dynamics

- The basic object is function
- Example
  - Protein interactions
  - Language

Functional dynamics:

$$f_{n+1}(x) = F(f_n(x),\ f_n \circ f_n(x)).$$

Example

$$f_{n+1}(x) = (1 - \epsilon) f_n(x) + \epsilon f_n \circ f_n(x).$$

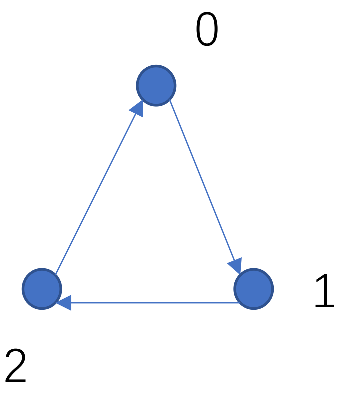
Example

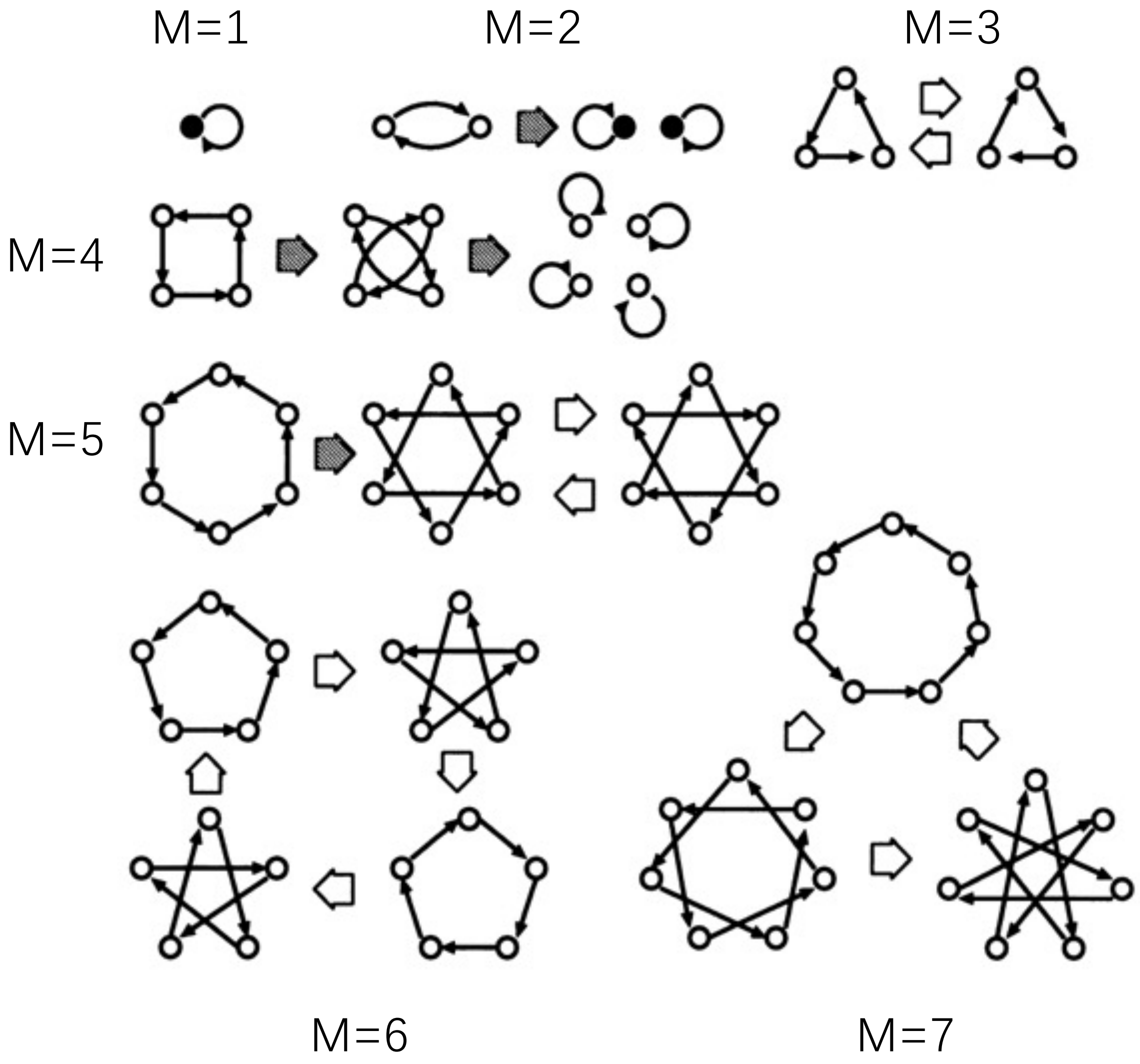
$$f_{n+1}(x) = f_n \circ f_n(x)$$

$$f_0(i) = (i+1)\ Mod\ M$$

Example: M=2

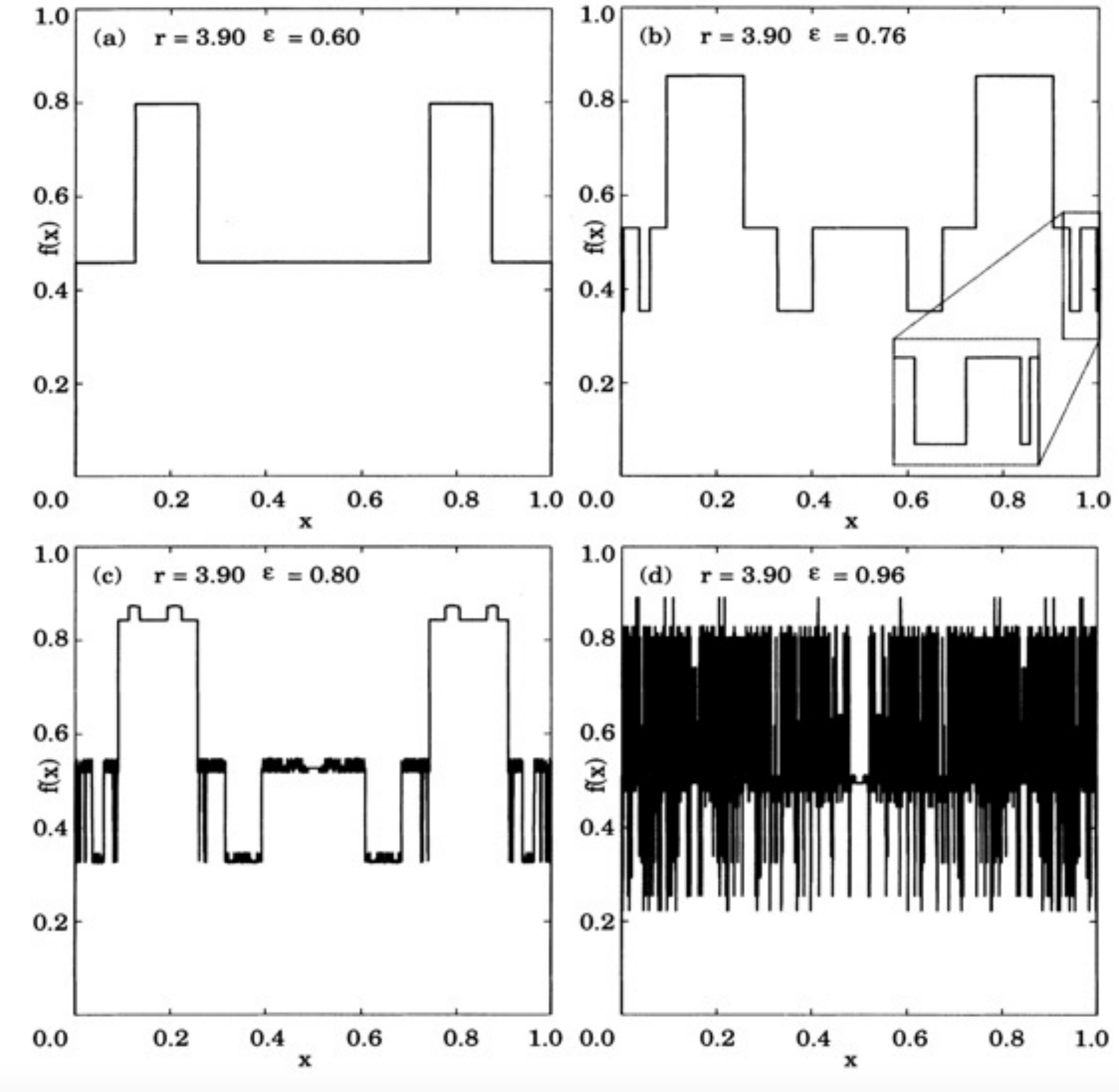
	0	1	2
0		1	
1			1
2	1		



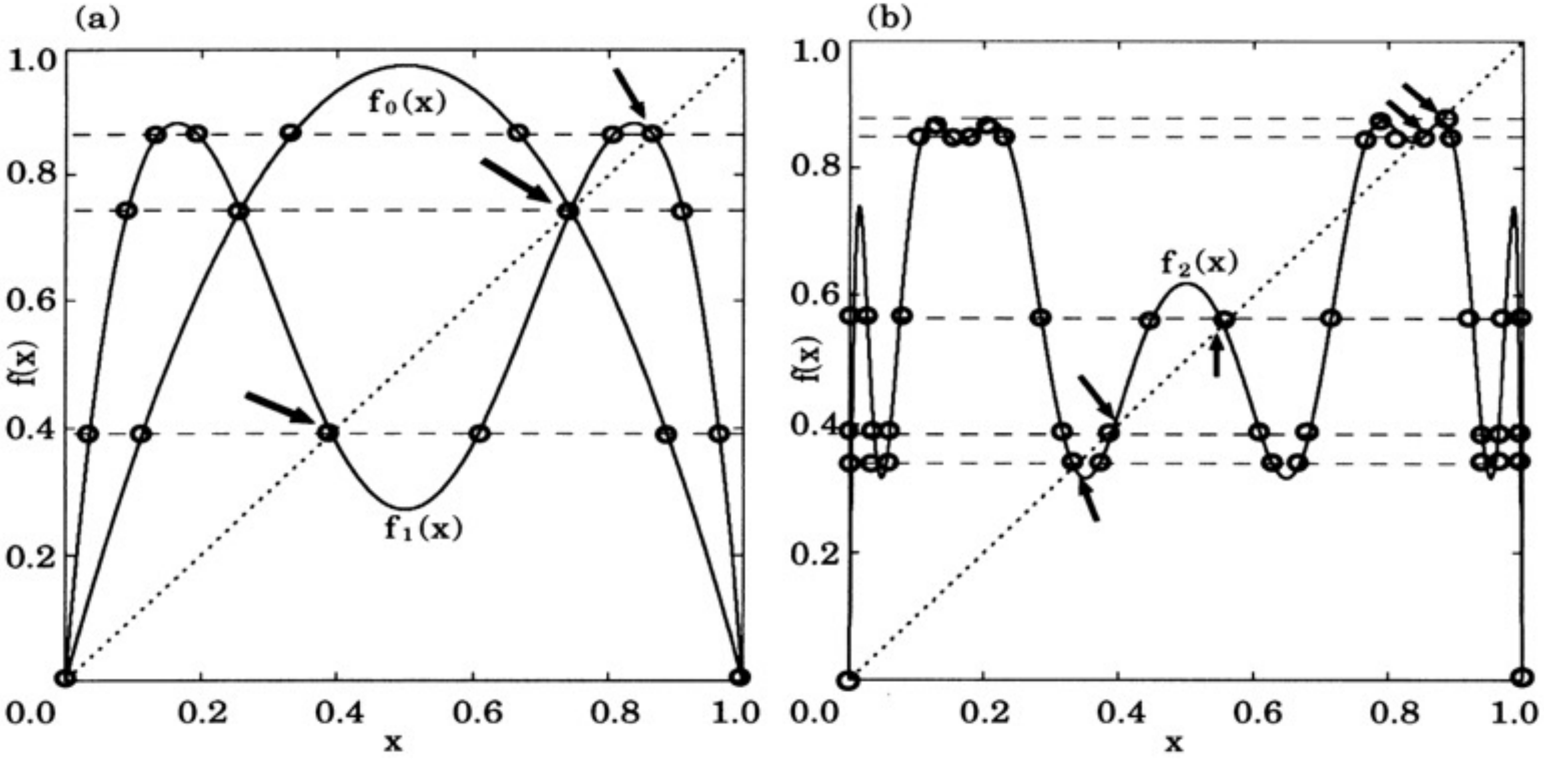


Four phases

- (S): Step phase
- (FS): Folded step phase
- (F): Fractal phase
- (R): Random phase



Two kinds of fixed points

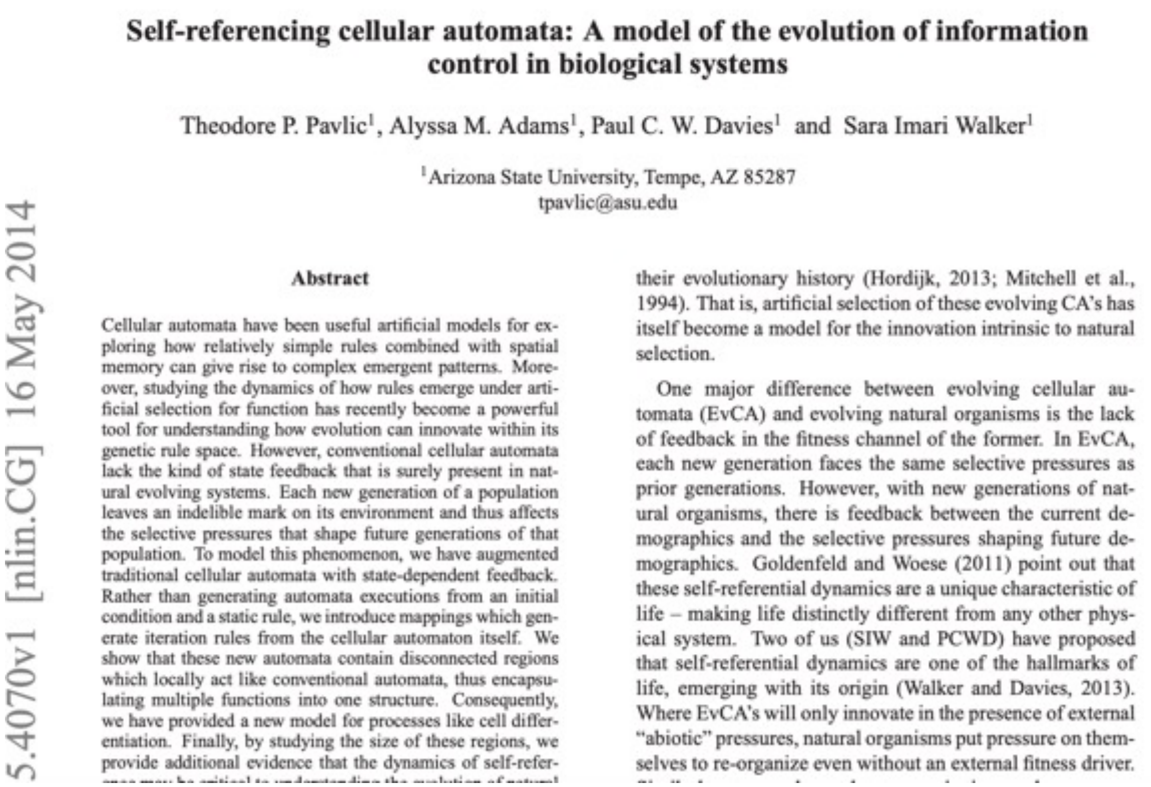


1. A point x^{I} satisfying $f(x^{\mathrm{I}}) = x^{\mathrm{I}}$ (type-I).
2. A point x^{II} satisfying $f(x^{\mathrm{II}}) = f \circ f(x^{\mathrm{II}})$ (but $f(x^{\mathrm{II}}) \neq x^{\mathrm{II}}$) (type-II).

Discussion

- Type I fixed point $f(x^{\mathrm{I}}) = x^{\mathrm{I}}$
  - a basis of symbolization in the abstract language space (x).
  - symbol for each articulated object.
  - The function as a filter articulates the continuous world x into a set of segments on each of which fn(x) assumes a distinct constant value.
- Type II fixed point $f(x^{\mathrm{II}}) = f \circ f(x^{\mathrm{II}})$
  - Self-referential conceptions
  - Artificial notions
- Fractal phase
  - Fine structures： fine conceptions

Self-referencing Cellular Automata



5.4070v1 [nlin.CG] 16 May 2014

Self-referencing cellular automata: A model of the evolution of information control in biological systems

Theodore P. Pavlic[1], Alyssa M. Adams[1], Paul C. W. Davies[1] and Sara Imari Walker[1]

[1]Arizona State University, Tempe, AZ 85287
tpavlic@asu.edu

Abstract

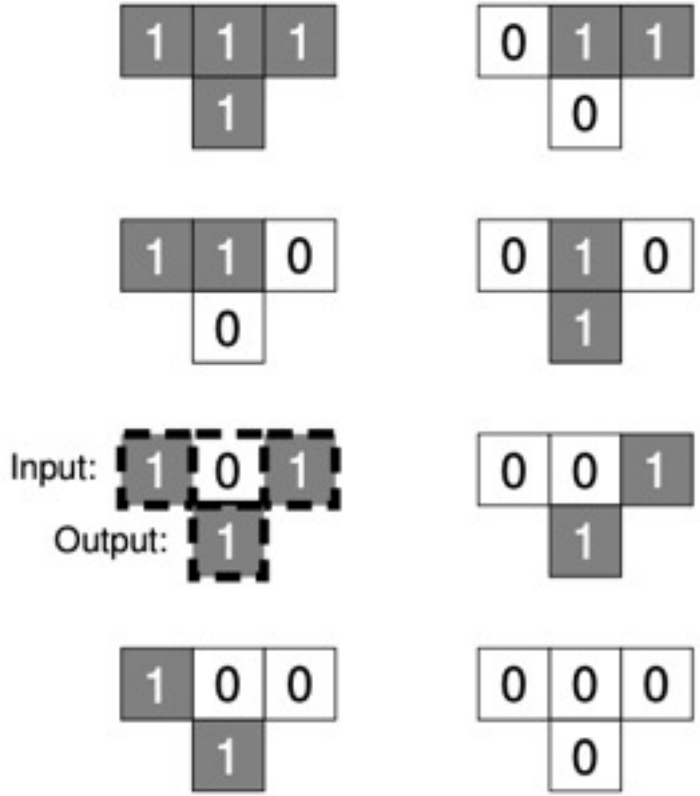
Cellular automata have been useful artificial models for exploring how relatively simple rules combined with spatial memory can give rise to complex emergent patterns. Moreover, studying the dynamics of how rules emerge under artificial selection for function has recently become a powerful tool for understanding how evolution can innovate within its genetic rule space. However, conventional cellular automata lack the kind of state feedback that is surely present in natural evolving systems. Each new generation of a population leaves an indelible mark on its environment and thus affects the selective pressures that shape future generations of that population. To model this phenomenon, we have augmented traditional cellular automata with state-dependent feedback. Rather than generating automata executions from an initial condition and a static rule, we introduce mappings which generate iteration rules from the cellular automaton itself. We show that these new automata contain disconnected regions which locally act like conventional automata, thus encapsulating multiple functions into one structure. Consequently, we have provided a new model for processes like cell differentiation. Finally, by studying the size of these regions, we provide additional evidence that the dynamics of self-refer-

their evolutionary history (Hordijk, 2013; Mitchell et al., 1994). That is, artificial selection of these evolving CA's has itself become a model for the innovation intrinsic to natural selection.

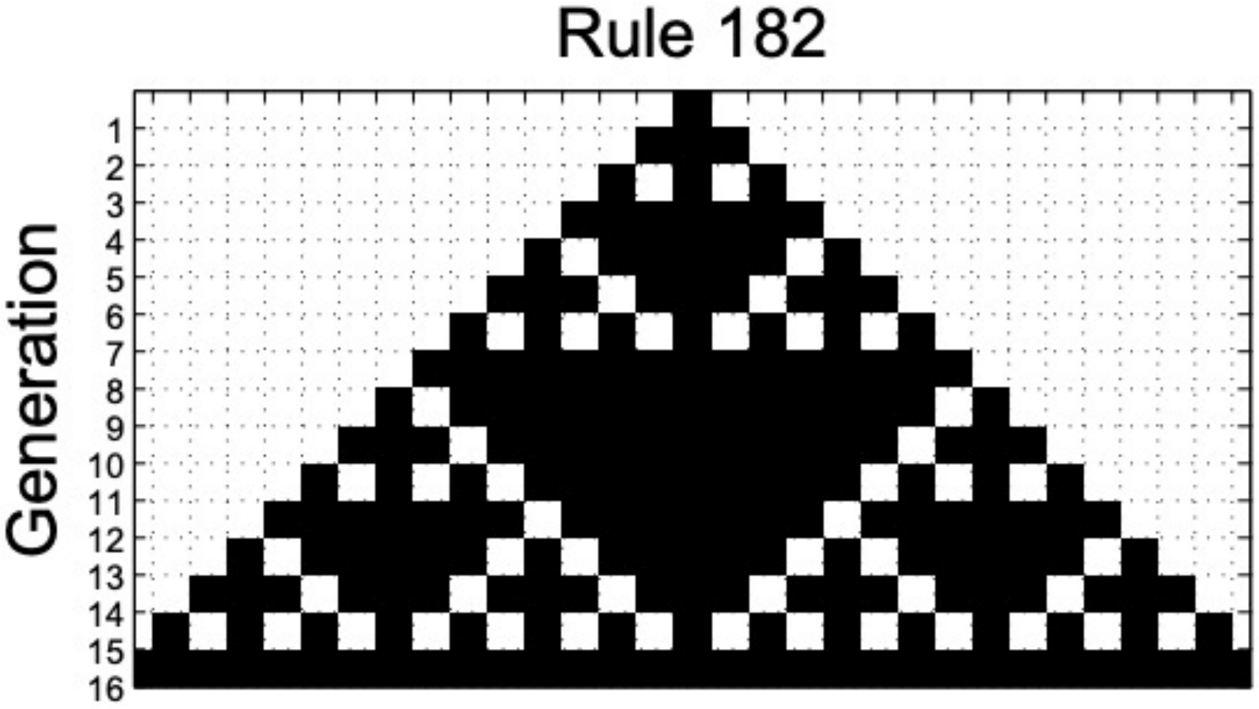
One major difference between evolving cellular automata (EvCA) and evolving natural organisms is the lack of feedback in the fitness channel of the former. In EvCA, each new generation faces the same selective pressures as prior generations. However, with new generations of natural organisms, there is feedback between the current demographics and the selective pressures shaping future demographics. Goldenfeld and Woese (2011) point out that these self-referential dynamics are a unique characteristic of life – making life distinctly different from any other physical system. Two of us (SIW and PCWD) have proposed that self-referential dynamics are one of the hallmarks of life, emerging with its origin (Walker and Davies, 2013). Where EvCA's will only innovate in the presence of external "abiotic" pressures, natural organisms put pressure on themselves to re-organize even without an external fitness driver.

拉里萨·阿尔班塔基斯（Larissa Albantakis），威斯康星大学麦迪逊分校的理论神经科学家

Cellular Automata

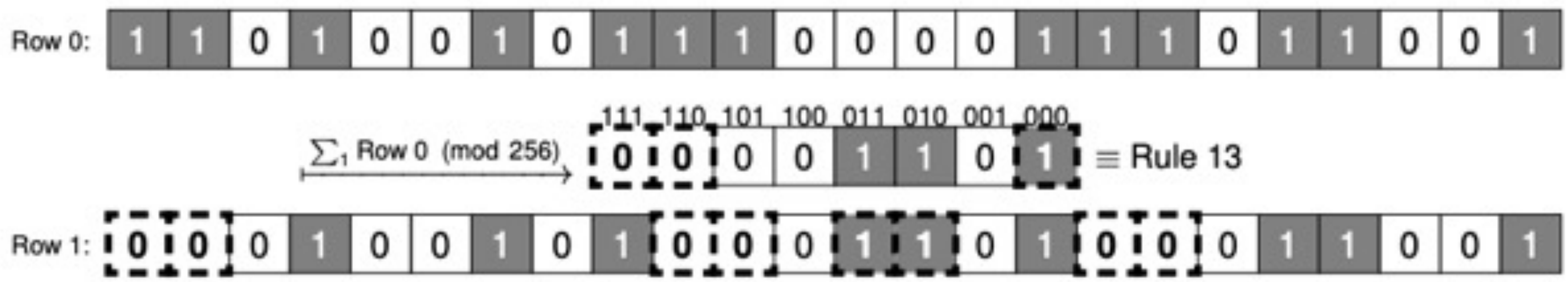


(a) Single iteration of rule 182



(b) Execution of rule 182

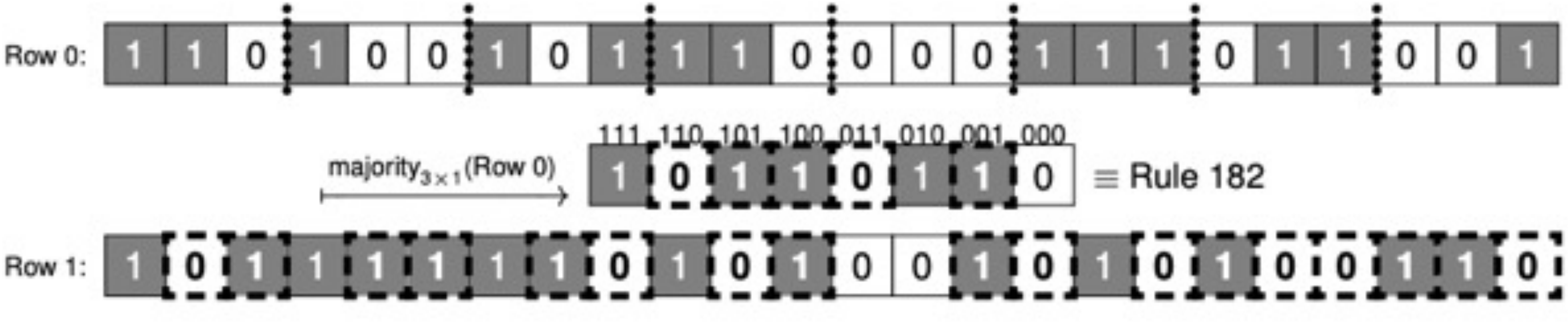
Self-referential Cellular Automata


Row 0: 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 1 1 0 0 1
111 110 101 100 011 010 001 000
$\sum_1$ Row 0 (mod 256)
0 0 0 0 1 1 0 1
≡ Rule 13
Row 1: 0 0 0 1 0 0 1 0 1 0 0 0 1 1 0 1 0 0 0 1 1 0 0 1

(a) Sum of ones

$$s(t) \rightarrow S(t) \rightarrow f \rightarrow s(t+1) = f(s(t))$$

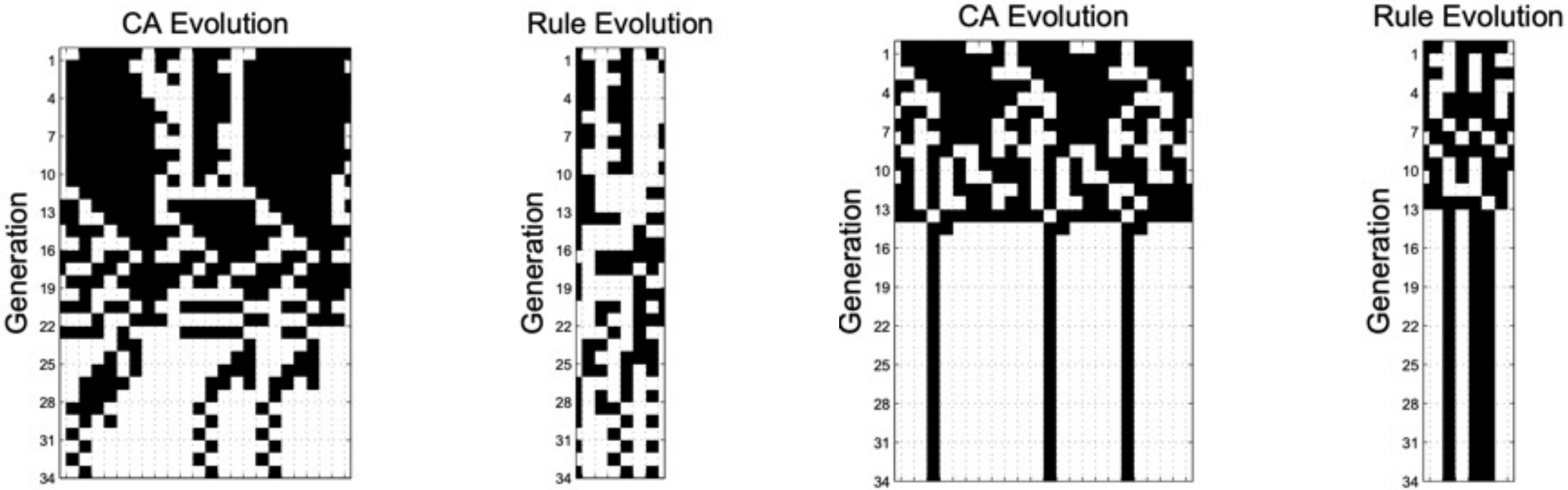
Self-referential Cellular Automata



(a) Block ones majority

$$s(t) \rightarrow S(t) \rightarrow f \rightarrow s(t+1) = f(s(t))$$

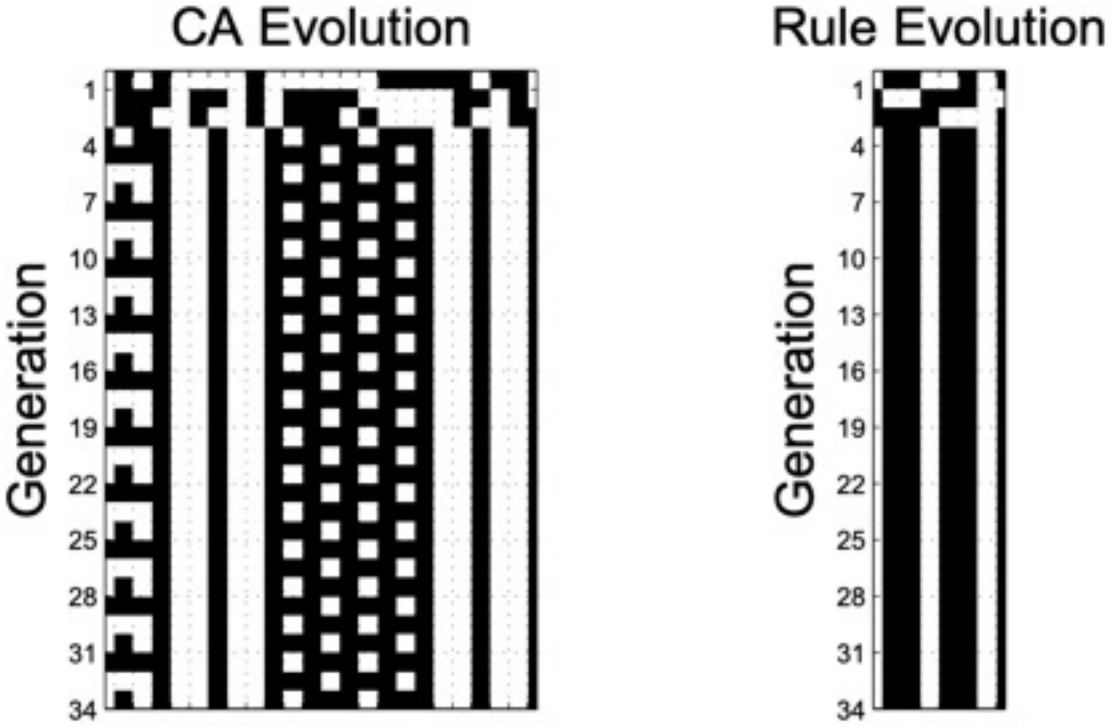
Results



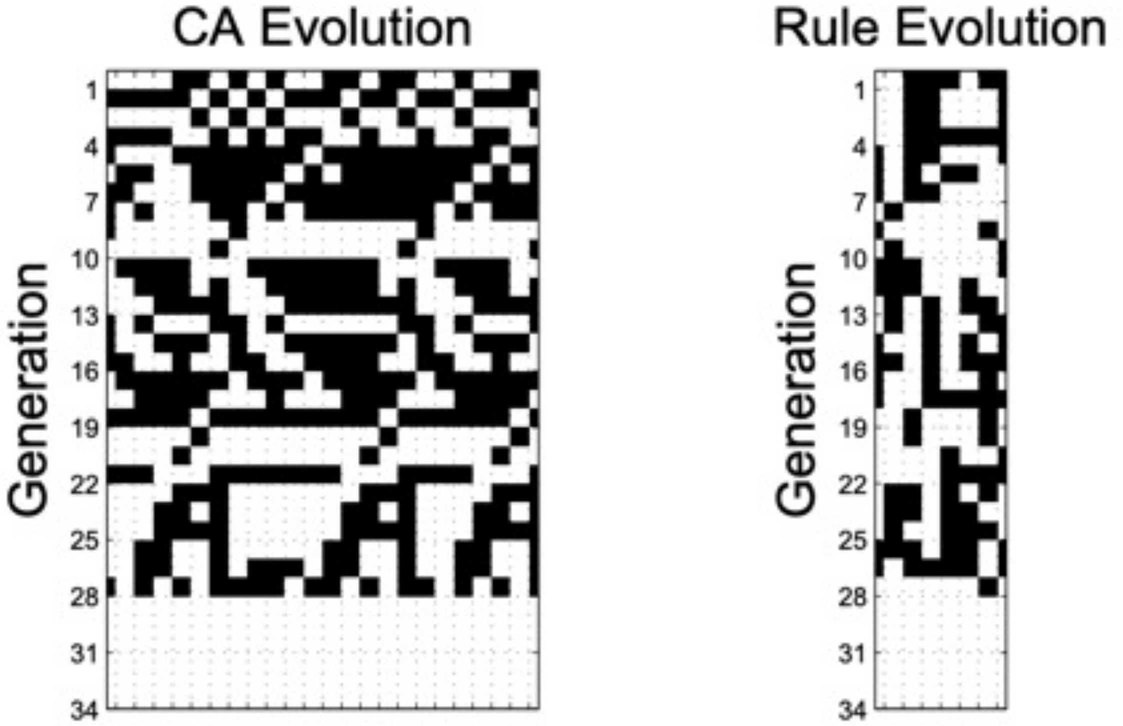
(a) Execution leading to mutual oscillation

(b) Execution leading to mutual fixed points

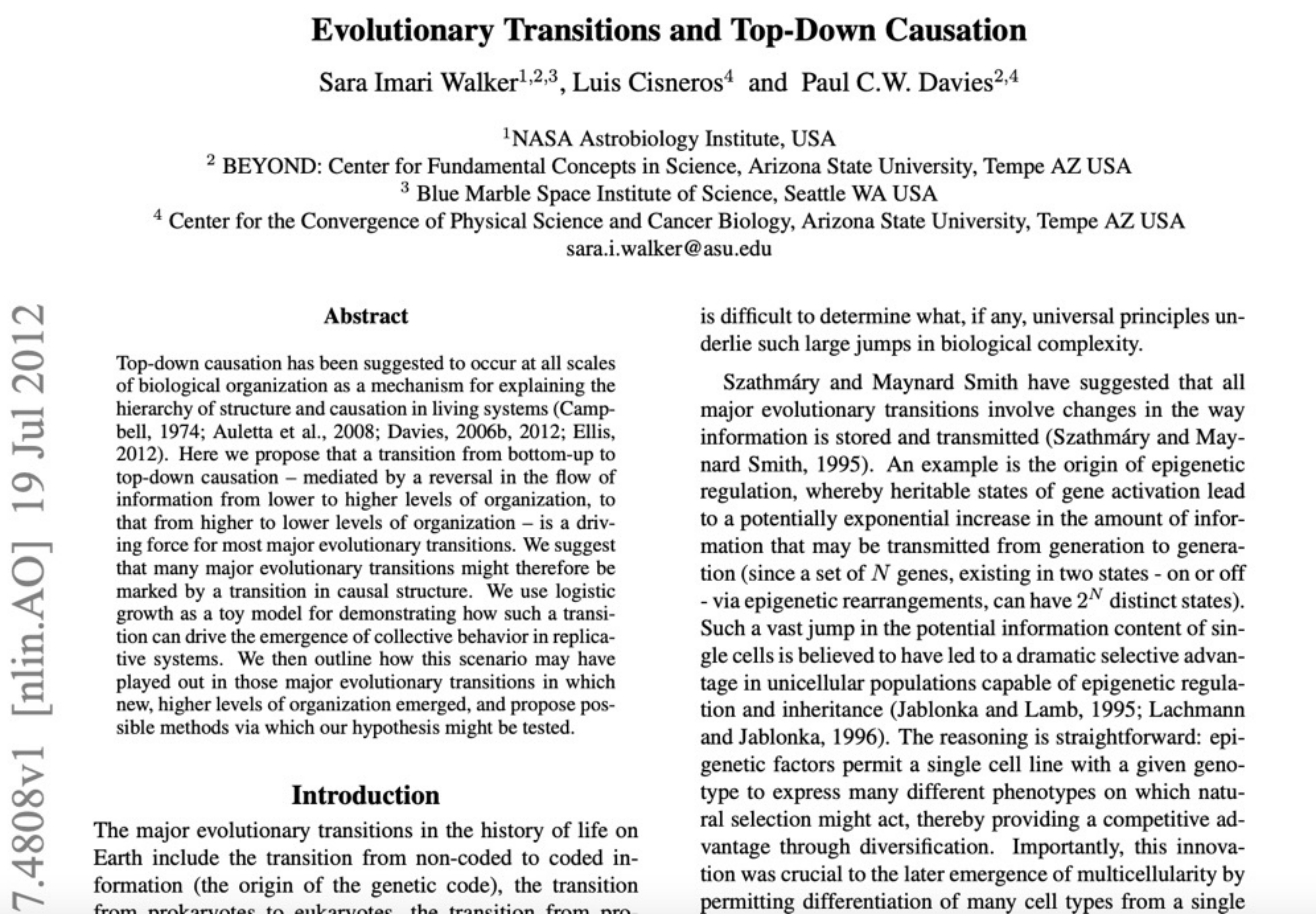
Results



Period-6 macroexecution with short transient



Fixed point with long structured transient



Evolutionary Transitions and Top-Down Causation

Sara Imari Walker[1,2,3], Luis Cisneros[4] and Paul C.W. Davies[2,4]

[1]NASA Astrobiology Institute, USA
[2] BEYOND: Center for Fundamental Concepts in Science, Arizona State University, Tempe AZ USA
[3] Blue Marble Space Institute of Science, Seattle WA USA
[4] Center for the Convergence of Physical Science and Cancer Biology, Arizona State University, Tempe AZ USA
sara.i.walker@asu.edu

Abstract

Top-down causation has been suggested to occur at all scales of biological organization as a mechanism for explaining the hierarchy of structure and causation in living systems (Campbell, 1974; Auletta et al., 2008; Davies, 2006b, 2012; Ellis, 2012). Here we propose that a transition from bottom-up to top-down causation – mediated by a reversal in the flow of information from lower to higher levels of organization, to that from higher to lower levels of organization – is a driving force for most major evolutionary transitions. We suggest that many major evolutionary transitions might therefore be marked by a transition in causal structure. We use logistic growth as a toy model for demonstrating how such a transition can drive the emergence of collective behavior in replicative systems. We then outline how this scenario may have played out in those major evolutionary transitions in which new, higher levels of organization emerged, and propose possible methods via which our hypothesis might be tested.

Introduction

The major evolutionary transitions in the history of life on Earth include the transition from non-coded to coded information (the origin of the genetic code), the transition from prokaryotes to eukaryotes, the transition from pro- is difficult to determine what, if any, universal principles underlie such large jumps in biological complexity.

Szathmáry and Maynard Smith have suggested that all major evolutionary transitions involve changes in the way information is stored and transmitted (Szathmáry and Maynard Smith, 1995). An example is the origin of epigenetic regulation, whereby heritable states of gene activation lead to a potentially exponential increase in the amount of information that may be transmitted from generation to generation (since a set of N genes, existing in two states - on or off - via epigenetic rearrangements, can have 2^N distinct states). Such a vast jump in the potential information content of single cells is believed to have led to a dramatic selective advantage in unicellular populations capable of epigenetic regulation and inheritance (Jablonka and Lamb, 1995; Lachmann and Jablonka, 1996). The reasoning is straightforward: epigenetic factors permit a single cell line with a given genotype to express many different phenotypes on which natural selection might act, thereby providing a competitive advantage through diversification. Importantly, this innovation was crucial to the later emergence of multicellularity by permitting differentiation of many cell types from a single

Top down causation

$$x_{i,n+1} = (1-\epsilon) f_i(x_{i,n}) + \epsilon\, m_n \quad ; \quad (i = 1, 2, \ldots N) \quad \textbf{(1)}$$

Top down interaction

Strength of coupling of top down interaction

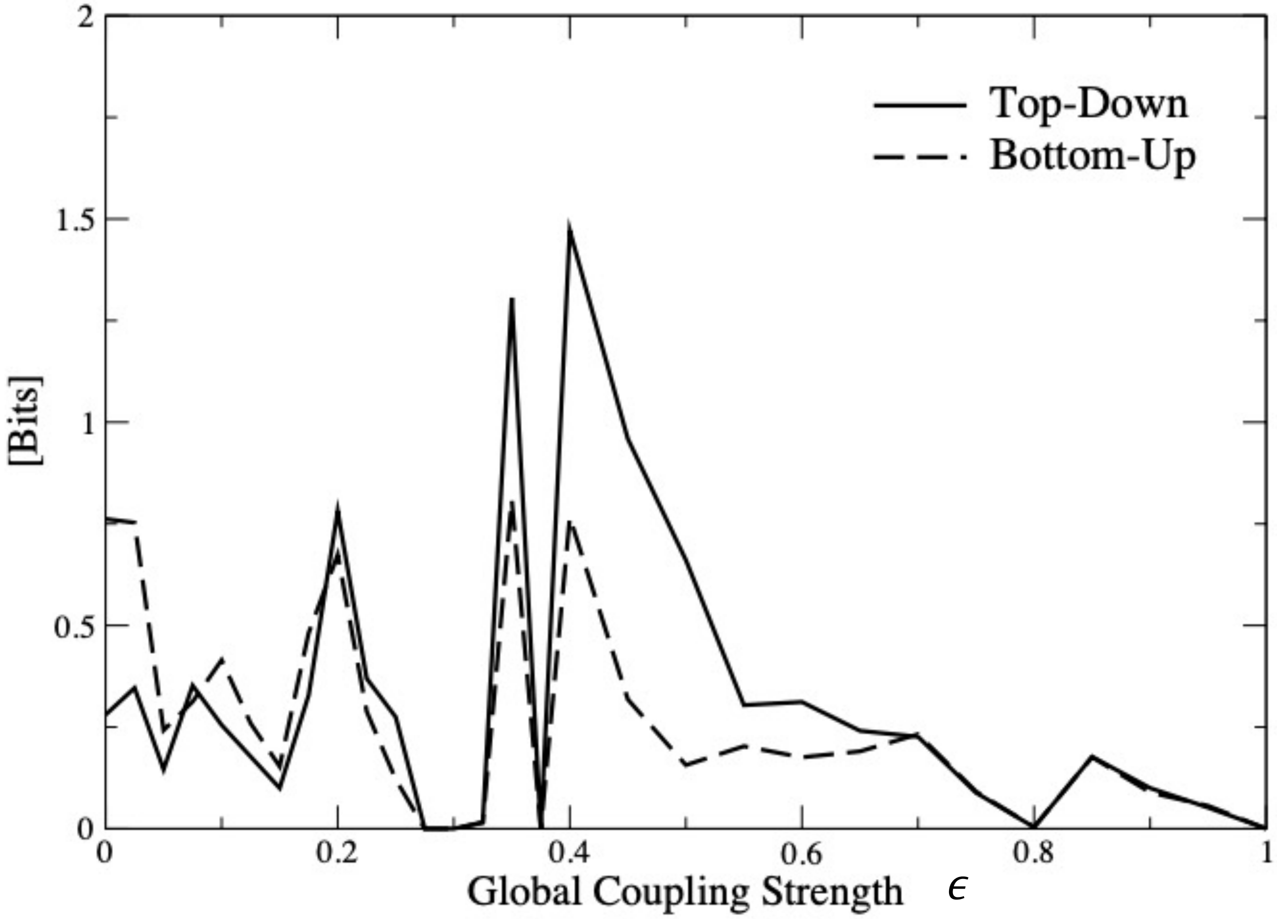
$$m_n = \frac{1}{N} \sum_{j=1}^{N} f_j(x_{j,n})$$

$$T^{(k)}_{Y \to X} = \sum_n p(x_{n+1}, x_n^{(k)}, y_n^{(k)}) \log \left[\frac{p(x_{n+1} | x_n^{(k)}, y_n^{(k)})}{p(x_{n+1} | x_n^{(k)})} \right]$$

$$f_i(x_{i,n}) = r_i x_{i,n} \left(1 - \frac{x_{i,n}}{K}\right)$$

Measure of causal force

When top-down causality emerge



Summary

- Emergence of causality
    - Top-down(formal) causality
    - Purpose(final) causality
    - Self-reference
- Self-reference in computation theory
    - Turing machine and TM-computable functions
    - S-m-n theorem and Universal TM
    - Self-referential paradox
    - Recursion Theorem
    - General self-reference in category theory
- Self-referential dynamics
    - Functional dynamics
    - Self-referencing Cellular Automata
    - Top-down causality in a collective logistic mapping