

# Welcome to

PHENOMENOLOGY

**COGNITION**

COMPUTATION

Dropbox-Ordner: PhenCoCo, Mail an [ael@viom.net](mailto:ael@viom.net)

## **For a short topic recap:**

PhenCoCo provides a circle of people who work out arguments for and against a computational foundation and the counterpart, embodiment, situated cognition, etc. Our members have different backgrounds and strengths, which will come in handy during our discussions and talks.

One goal is to combine approaches, when possible, to avoid methodologically transfigured conclusions and arrive at a thesis, which can be published.

Another goal is to present the results of our work in form of a symposium, which will include speakers outside of the group. Hopefully, this will diversify and maybe draw a connection to practice.

We will meet once a month at the Humboldt University and try to make it possible for everyone to attend, mostly by using doodle to compare schedules. All this is still in discussion.

## Agenda:

### Organisational Introduction

- Date of monthly meeting
- Info: Funding for symposium

### Content Introduction

- RECAP topic introduction
- Presentation: AEL “A Computational Foundation for the Study of Cognition” by David Chalmers

### Collective discussion of Strategy

- Details about texts to read
- Way stones (essays & presentation for the symposium)

# A Computational Foundation for the Study of Cognition

David J. Chalmers

Vortragende: A.-E Lipinski

Aim of the Paper: Justifying the **role of computation**  
as true foundation for modern cognitive science,  
by advocating a kind of minimal computationalism.

Structure:

First Step The role of computation is clarified by  
the **analysis or theory of implementation (ToI)**.  
(A system implements a computation if the causal structure of the  
system mirrors the formal structure of the computation.)

Second Step The **ToI** justifies the central commitments to **AI & CCS ...**

ToI justifies Commitments to:

**Artificial intelligence (AI):**

and

**Computational cognitive science (CCS):**

*The thesis of computational sufficiency  
(The right kind of computational structure  
suffices for the possession of a mind.)*

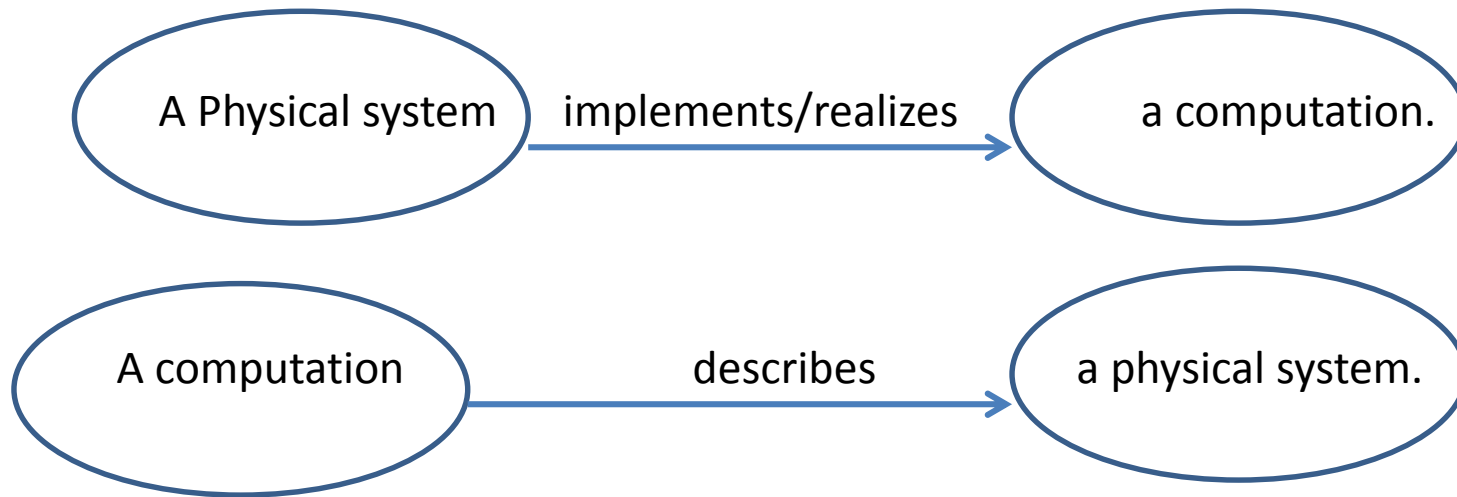
*The thesis of computational explanation  
(Computation provides a general framework  
for the explanation of cognitive processes.)*

Two 'facts' which support the claims (1) and (2),  
later in the Text a lot is build on them:

a) Computation can **specify** general  
patterns of causal organization

b) Mentality is an **organizational invariant**,  
rooted in general patterns of causal  
organization

## Clarifying the notion of computation:



A **bridge** is required between computational theory and implementing system.

# Clarifying the notion of computation:

1<sup>st</sup> Question:

What are the conditions under which a physical system implements a given computation?

Searle: ... any given system can be seen to implement any computation if interpreted appropriately.

(Wordstar example: A wall can be described as highly complex system that consist of many different states and therefore can seen as an implementation of a random program.

& later on: digestion example)

Chalmers: ... there are objective conditions for implementation that can be spelled out.

See theory of implementation



# Justifying AI and CCS :

## 2<sup>nd</sup> Question:

What is the relationship between computation and cognition?

Chalmers: ...the properties of a physical cognitive system that are relevant to its implementing certain computations, [...] are precisely those properties in virtue of which

- a) the system possesses mental properties
- b) the system's cognitive processes can be explained

The computational framework developed in question 1 ...

- ... can *therefore* be used to justify the theses of computational sufficiency\* and computational explanation\*\*.
- ... will be used to answer various challenges to the centrality of computation, which will show the solidity of the foundations of artificial intelligence\* and computational cognitive science\*\*.

## Justifying AI and CCS :

The computational framework developed in question 1 ...

- ... can *therefore* be used to justify **the theses of computational sufficiency** and **computational explanation**.
- ... will be used to answer various challenges to the centrality of computation, which will show the solidity of the foundations of artificial intelligence and computational cognitive science.

# A Theory of Implementation (ToI):

## The Answer to the 1<sup>st</sup> question:

(What are the conditions under which a physical system implements a given computation?)

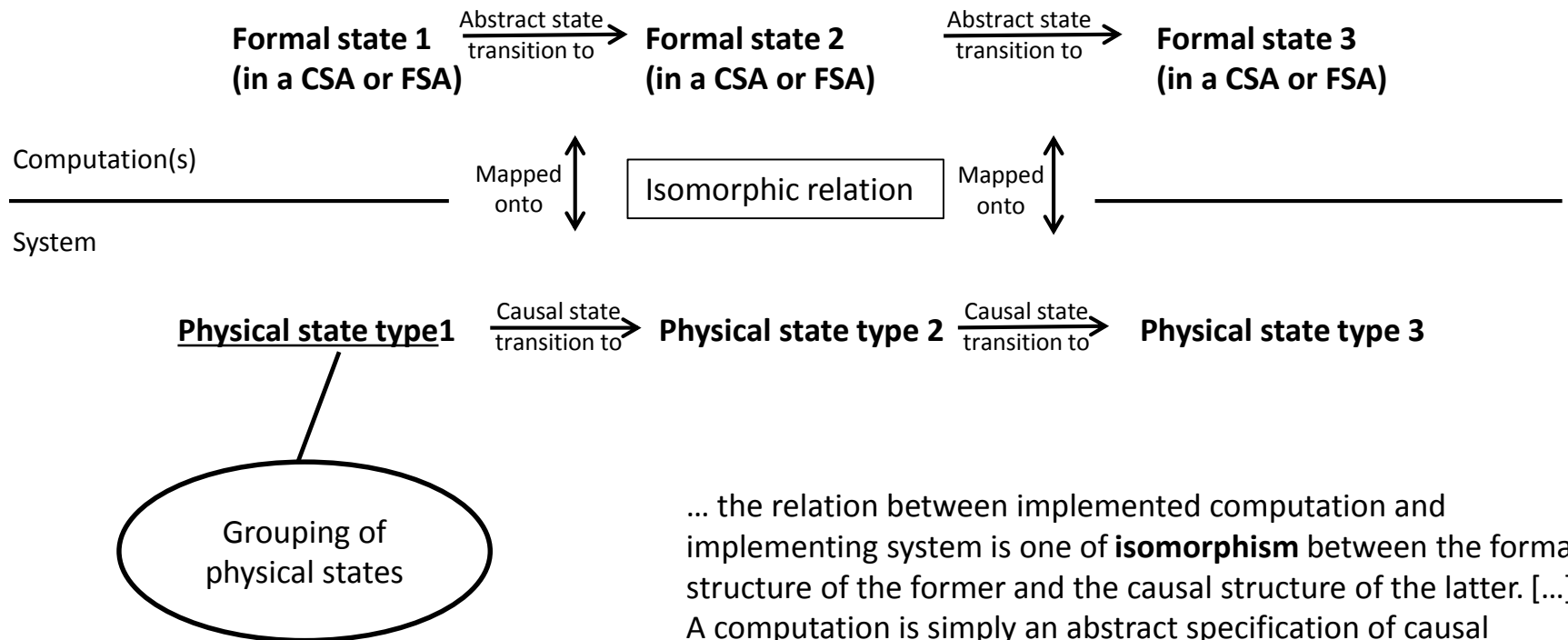
A physical system implements a given computation when

- there exists a **grouping** of physical states of the system into **state types**
- there is a one-to-one **mapping** from **formal states** of the computation to **physical state types**  
(such that formal states related by an *abstract state transition*  
are mapped onto  
physical state types *related* by a corresponding *causal state transition*)

Specification of the class of computations in question:

- Computations are specified relative to some formalism
- The variety of formalisms can be subsumed under the class of **combinatorial-state automata (CSA's)**  
[Exapmls for CSA's/ formalisms: Pascal programs, cellular automata, neural networks, Turing machines]  
(illustrated by the case of **simple fine-state relata (FSA's)**)

# A Theory of Implementation:



... the relation between implemented computation and implementing system is one of **isomorphism** between the formal structure of the former and the causal structure of the latter. [...] A computation is simply an abstract specification of causal organization. (S. 6)

# Simple fine-state relata/ automata

Sets given by the FSA:

Input states:  $I_1, I_2, \dots, I_k$

Internal States:  $S_1, S_2, \dots, S_m$

Output states:  $O_1, O_2, \dots, O_n$

State-transition relations given by the FSA:

$(S, I) \rightarrow (S', O')$  (For each pair  $(S, I)$  where  $S'$  and  $O'$  are an internal state and an output state respectively.)

$(S_0, I_0$

$S_n, I_n) \rightarrow (S_{n+1}, I_{n+1})$

( $S$  &  $I$  can be thought of as the “old” internal state and the input at a given time.

$S'$  is the “new” internal state and  $O'$  is the output produced at that time.)

Does the output happen at the same time as the input?  
→

*Variations of this being spelled out, e.g. outputs are not necessarily included in every step.*

## Simple fine-state relata/ automata (FSA)

A physical system  $\mathbf{P}$  *implements* an FSA  $\mathbf{M}$

- if there is a mapping  $\mathbf{f}$  that *maps* internal states of  $\mathbf{P}$  to internal states of  $\mathbf{M}$
- and a mapping  $\mathbf{f}$  that *maps* output states of  $\mathbf{P}$  to output states of  $\mathbf{M}$ :

For every state-transition  $(S, I) \rightarrow (S', O')$  of  $\mathbf{M}$ , the following *conditional* holds:

- If  $\mathbf{P}$  is an internal state  $\mathbf{s}$  and receiving input  $\mathbf{i}$   
**where  $\mathbf{f}(\mathbf{s}) = \mathbf{S}$  and  $\mathbf{f}(\mathbf{i}) = \mathbf{I}$**
- $\mathbf{P}$  will be caused to enter internal state  $\mathbf{s}'$  and produce  $\mathbf{o}'$ ,  
**such that  $\mathbf{f}(\mathbf{s}') = \mathbf{S}'$  and  $\mathbf{f}(\mathbf{o}') = \mathbf{O}'$**

# Combinatorial state automata (CSA)

The patterns in most computational formalisms have a combinatorial structure:

- a cell pattern in a cellular automation
- a combination of tape-state and head-state in a Turing machine
- variables and registers in a Pascal program (programmers language)
- and other examples. (p.4)

Difference to FSA:

- An internal state is specified by a **vector** instead of a “monadic label  $S$ ”
- CSA's can implement complex computations more efficiently

## Questions answered

- Every system can implement some computation
- Not every system can implement any given computation  
(requirement: mapped states must satisfy reliable state-transition rules)
- A given system can implement more than one computation  
(In general, there is no canonical mapping from a physical object to “the” computation it’s performing. We might say that within every physical system, there are numerous computational systems.)  
→ Example: A Computer running more than one program



## Questions answered

- If every process implements some computation, what is **special about cognition**?

A system's implementation of a computation is **not necessarily bound to a certain physical instance**, like digestion, it could turn out to be something different.

- Not every computation is digestion, obviously, digestions is (some) computation.

## Questions answered

### Putnam's argument:

With this definition, almost any physical system can be seen to implement every fine-state automaton. (Within a timeframe of, say, 10 minutes)

→ Mapping from physical states of a system to internal states of an FSA, under the assumption that physical states do not repeat:

- A is always followed by B, in a certain time period
- Inputless FSA: The initial physical state is mapped to the initial formal state, and the successive states are mapped onto each other


## Questions answered

Chalmers' answer:

- 'A is followed by B' is not the only requirement
- Inputless FSA's are not appropriate for mind modulation, due to their non-combinatorial structure
- Such a conditional must be satisfied for **every transition** in the machine table, not only for those in one time period
- There must be a reliable **counterfactual-supporting connection** between the states  
(if a system is in state A it will transit to state B)

Objection (phencoco): The given time period is non-analogous to cognitive processes, who can be unfinished. Also FSA cannot neither simulate nor modulate this infinite regress.

## Questions answered:

 ... there is a class of computations such as any system implementing that computation is a cognitive system

(and possibly every cognitive system implements a kind of computation that is itself always cognitive & shares mental properties with the original system)

## Questions answered: What about semantics?

- Computations are specified syntactically, not semantically

Semantic considerations within the conditions of implementation would endanger any role of computation in giving ground to AI and CCS,

**because the notion of semantic content lacks foundation itself**

- The notion of Semantic content does not depend on the notion of computation
  - Computation and semantic content should depend on 'the common pillar' of causation
- By keeping the notions combined thusly, they do not codepend artificially but do not count wrongly as unrelated either

# Computation and cognition

- Does computationally produced behaviour suffice for mind?

(Common approach: Appeal the Turing Test

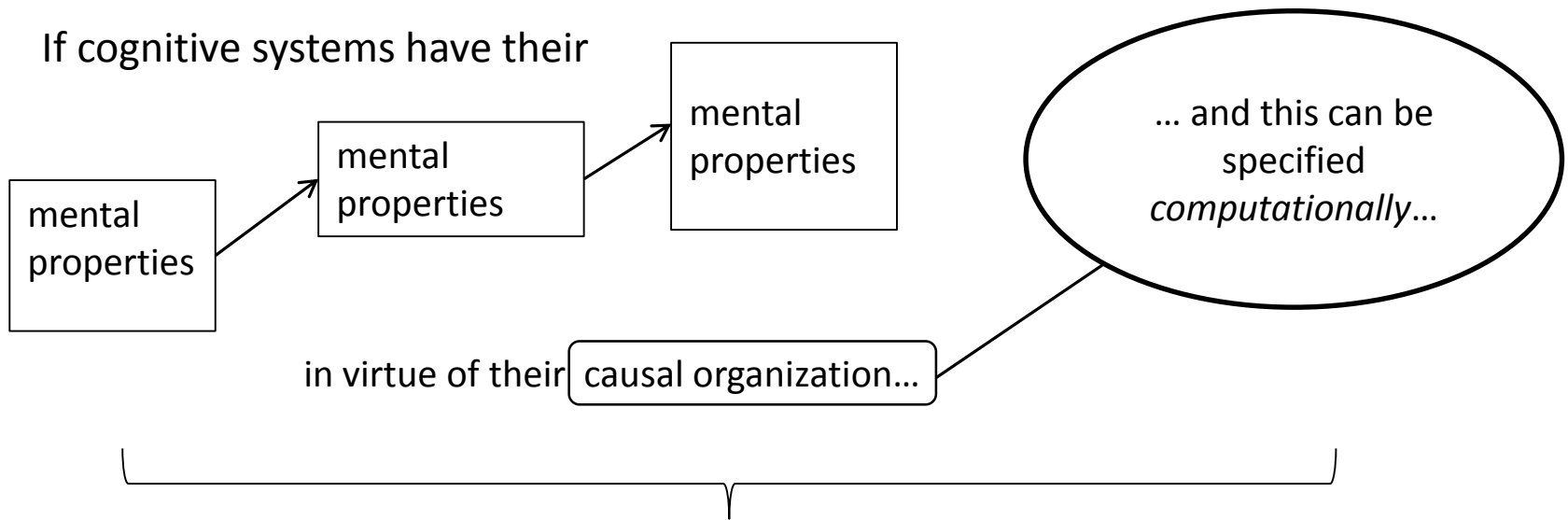
Every implementation of a given computation will have

**a certain kind of behaviour,** and this behaviour is sufficient for mentality.

- Chalmers: The demise of logical behaviourism has made it implausible, that **this behaviour** suffices for specific mental properties. Two distinct mental systems can share the same behavioural dispositions. A computational basis for cognition requires a tighter link

# Computation and cognition

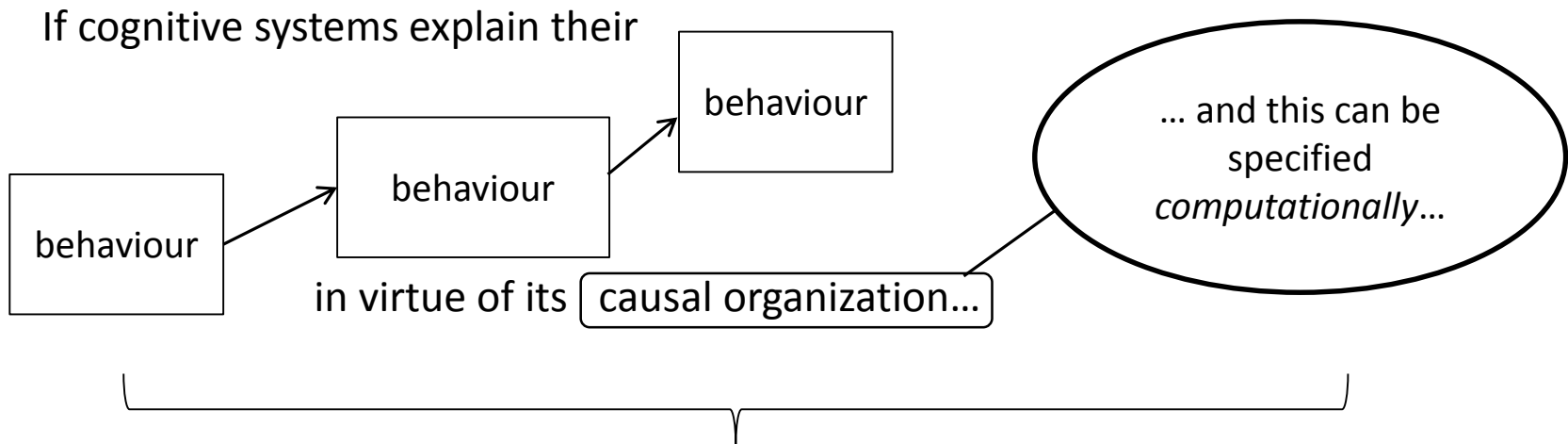
- Central property of computation:  
The ability to provide an abstract specification of the causal organisation of a system



...then the thesis of **computational sufficiency** is established.

# Computation and cognition

- Analogy to the thesis of computational explanation:



...then the thesis of **computational explanation** is established.



## Computation and cognition

### Objection (phencoco)

- The description of behaviour as made major contributions to the development of complex algorithms, which lead to the idea it might be possible to construct intelligence this way too.
- Chalmers seems to take the opposite approach and tries to deduce the computability of cognition from these algorithms

# Organizational invariance

- Introduction of the notion of **causal topology**  
(The pattern of interaction among parts of the system)
- Any system will have causal topology at a number of different levels
- For cognitive systems the level of causal topology will be fine enough to determine the causation of behaviour
- For the brain, the causal topology will probably be found on the neural level or higher

# Organizational invariance

Property P = organizational invariant

(A change to the system that preserves the causal topology, preserves P)

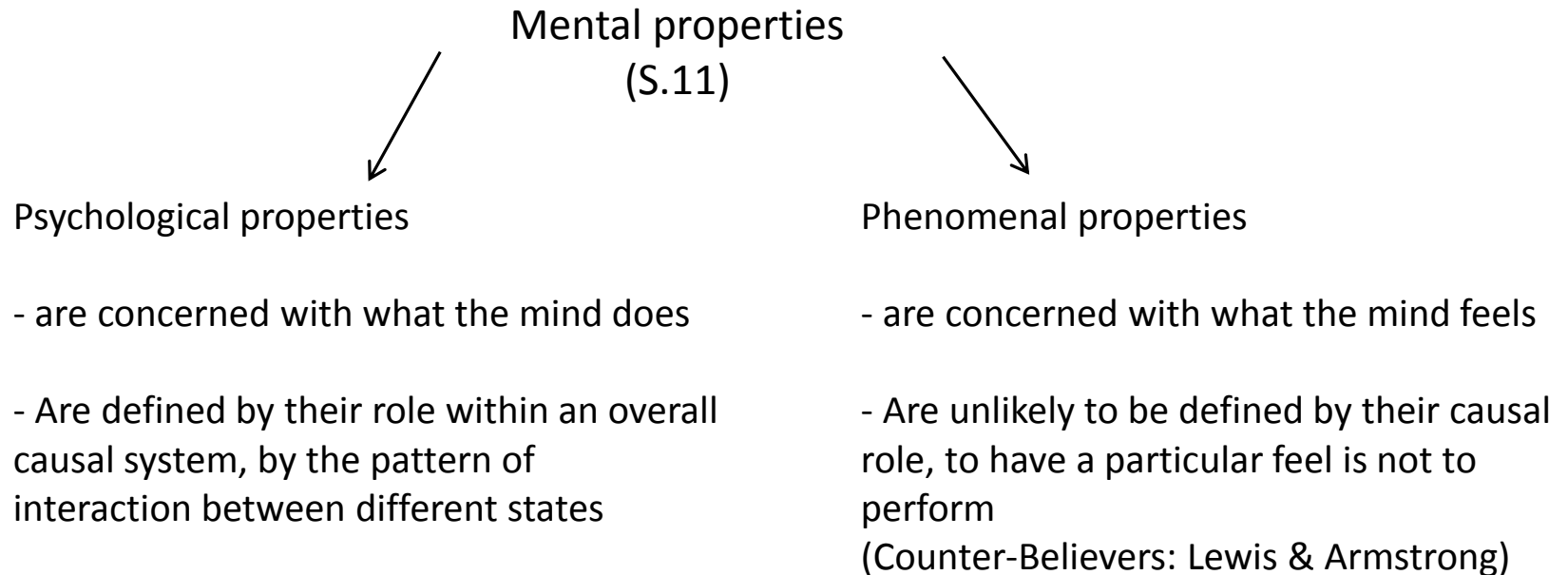
→ Most properties depend essentially on features that are not features of causal topology  
if these features are changed, the properties will change.

Examples: flying, digestion, being a tube of toothpaste (S. 10)

# The organizational invariance of mental properties

- Most mental properties are organizational invariants
- Not those that are partly supervenient on states of the environment
- Mental properties that only depend on internal (brain) states "...will be organizational invariants" (S.10)

# The organizational invariance of mental properties



“Dancing Qualia” argument (S.12):

Mental properties must be organizational invariants, it would be absurd if they weren't.

# Justifying the thesis of computational sufficiency & explanation

- A Pattern of the causal interaction between parts of the system
- Can be abstracted into a CSA description,  
by parts of the system corresponding to elements of the CSA state-vector  
and the patterns of interaction expressed in the state-transition rules  
(Requirement: each part has only a finite number of states that are relevant to the  
causal dependencies between the parts)

# Justifying the thesis of computational sufficiency & explanation

- If the above is correct the following is established:
  - the thesis of computational theory
  - The view called “strong artificial intelligence” (Searle)
- 🔗 There exists some computation such that any implementation of the computation possesses mentality  
(... and possibly every cognitive system implements a kind of computation that is itself always cognitive & shares mental properties with the original system)
- Analogy to the thesis of computational explanation

# The organizational invariance of mental properties

Central claim (S.10):

Most mental properties are organizational invariants.

- Exceptions: properties that are partly supervenient on states of the environment.
- Examples: knowledge, belief

Auxiliary claim:

Mental properties that depend only on internal (brain) state will be organizational invariants

- Reason: taking the environment's role into account



## Objections types (along the lines of:)

1) Computation cannot do what  
Cognition does (capacity)

2) Computation might capture the capacities,  
but still more is required for mentality

## Objection Tokens (OT's)

**OT1** Why should a simulation be counted as real mind?

→ If a computer model is appropriately designed it will share the causal topology of the system being modeled, the systems organizationally invariant properties (not only be simulated but) will be replicated.

It follows that a model that is computationally equivalent to a mind will itself be a mind

## Objection Tokens (OT's)

**OT2** Searle's Syntax and semantic objections (p. 14)

- 1) A computer program is syntactic
- 2) syntax is not sufficient for semantics
- 3) Minds have semantics; therefore
- 4) Implementing a computer program is insufficient for a mind

→ Programs and implementations are different:

- Programs are syntactical objects,
- implementations are physical systems with complex causal organization (!)

# Objection Tokens (OT's)

## OT3

Chinese Room: The homunculus does not understand, why would the system?  
A system possesses its mental properties in virtue of this causal organization.

Rerun of the “**dancing qualia**” argument (S.14)

Setup:

- Neurons are replaced by demons, step by step, until there is only one demon doing all the work of maintaining the causal organization
  - It is implausible to suppose that the system's experiences will change or disappear.
  - The system should not be confused with the demon itself, the demon does not experience anything
- Mental properties arising from distinct computational systems will be quite distinct, and there is no reason to suppose that they overlap
- (?)

# Objection Tokens (OT's)

## OT4

CSA formalisms capture discrete causal organisation, some mental properties may depend on continuous aspects like analog values or chaotic dependencies

→ Various substates of a CSA could be altered by an appropriate parameter for the transition rules and

## Other kinds of computationalism

Edelmann (1989):

- humans are not Turing machines and
- analysis of categorization procedures of humans and animals suggests the environment, not to be a tape for a Turing machine

Chalmer's answer:

- No one says brains are Turing machines, those are an example for implementation
- The computational framework is based on explaining and replicating human cognitive processes
- It may well turn out, that the causal organisation between individuals differs

## Other kinds of computationalism

Note: Computationalism is often associated with rule-following (S.17)

→ It is entirely possible that the only **rules** found in a computational description of thought will be at a very low level,

specifying the causal dynamics of neurons

OR

the dynamics between the neural and the cognitive

## Other kinds of computationalism

- **“strong” Computationalism: symbolic computationalism**  
(Newell, Symon, Fodor)
  - computational primitives in a computational description of cognition are representational primitives
  - state-transitions are defined between basic syntactic entities
  - Syntactic entities bear semantic content & are therefore *symbols*
- Smolemksy (1988): Connectionist systems perform **sybsymbolic computation**
  - The level of computation falls below the level of representation,
  - but the systems are computational nevertheless



## Other kinds of computationalism

- Legacy of the Turing machine

→ Chalmers: The notion of mechanism &  
The universality of its formalization

*"It is this univesaliyty that gives us good reason to suppose that computation can do almost anything that any mechanism can do, thus accounting for the centrality of computation in the study of cognition."  
(S.18)*

# Conclusion

## Minimal computationalism

- Is defined by the twin theses of **computational sufficiency** and **computational explanation**
- Computation taken in the broad sense that dates back to Turing
- Minimal computationalism is no bold empirical hypothesis but could still be refuted by  
e.g. proving the fact, that our cognitive capacities depend essentially on **infinite precision** in certain analogue qualities
- Computation is such a valuable tool, because it can describe almost any form of cognitive theory, even though the formalisms within may vary