What is Computation

- An Epistemic Approach –

Keynote

Jan van Leeuwen
Center for Philosophy of Computer Science
Utrecht University, NL

Jiří Wiedermann
Institute of Computer Science, Prague
Academy of Sciences of the Czech Republic

COMP UTAT ION

- 6th AISB Symp Computing & Philosophy, Exeter, 2013
- 7th AISB Symp. Computing & Philosophy, London, 2014
- Submitted papers, 2015

… how has our conception of *computation* evolved*?*

… what *is* computation?

Outline

1. The question
2. Philosophy at the rescue
3. Our view of computation
4. Computation as an observer-relative notion
5. A theoretical model*
6. Towards a computational theory of creativity*
7. Conclusion

1. The question

A.M. Turing (1912-1954)

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

Turing's broader view: how do processes work… computationally?

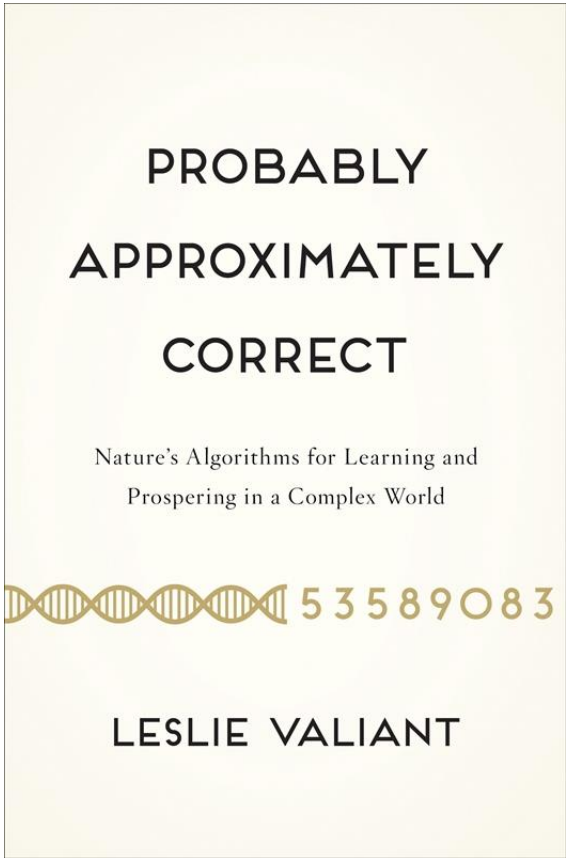
- Understanding computation (as machines…):
  - *Automatic machines* (a-machines, 1936) (*logical computing machines*, 1948)
  - Choice machines (c-machines, 1936)
  - With intuition: Oracle machines (o-machines, 1938)
  - Partially random machines (r-machines, 1948), machines random element (1950)
  - Modifiable machines incl. interference with machines (1950)
  - Practical computing machines (1948)
  - Continuous machines (1948)
  - No problem with potentially infinite computations and storage
- Understanding the brain:
  - Unorganized machines (1948)
    - Organized: A-type, B-type (artificial neural nets, brain model)
    - Self-organizing: P-type (learning nets)
    - Educating: learning machines
  - Intelligent machines, how they might `learn'
  - Machines that can play games (e.g. chess)
  - Can machines think (-> the imitation game aka the Turing test, 1950)
- Understanding (the algorithms of) nature…
  - Morphogenesis (1950 – 1954)

A.M. Turing (1951)

"My contention is that ***machines*** *can be constructed* ***which will simulate the behaviour of the human mind very closely***. They will make mistakes at times, and at times they may make new and very interesting statements, and on the whole the output of them will be worth attention to the same sort of extent as the output of a human mind."

- Understanding mental processes in terms of, or even *as*, **computation**?

Leslie Valiant (2013)



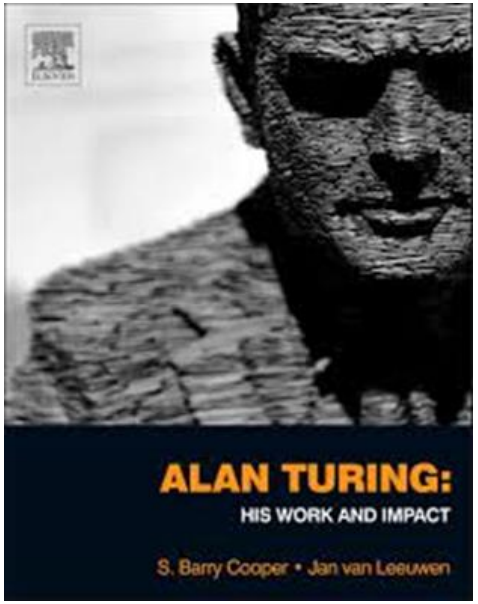
"Turing's concept [*of computation*] may enable us to understand human activity itself. […] even in the Pre-Turing era, in fact since the beginning of life, the dominating force on Earth within all its life forms *was* computation. [..] These computations were weak but they were exceedingly good, however, at one enterprise: adaptation. These are the computations that I call *ecorithms* […]

- Understanding (all) natural processes in terms of, or even *as*, **computation**?

Samson Abramsky (2013)

"*Two Puzzles about Computation*"

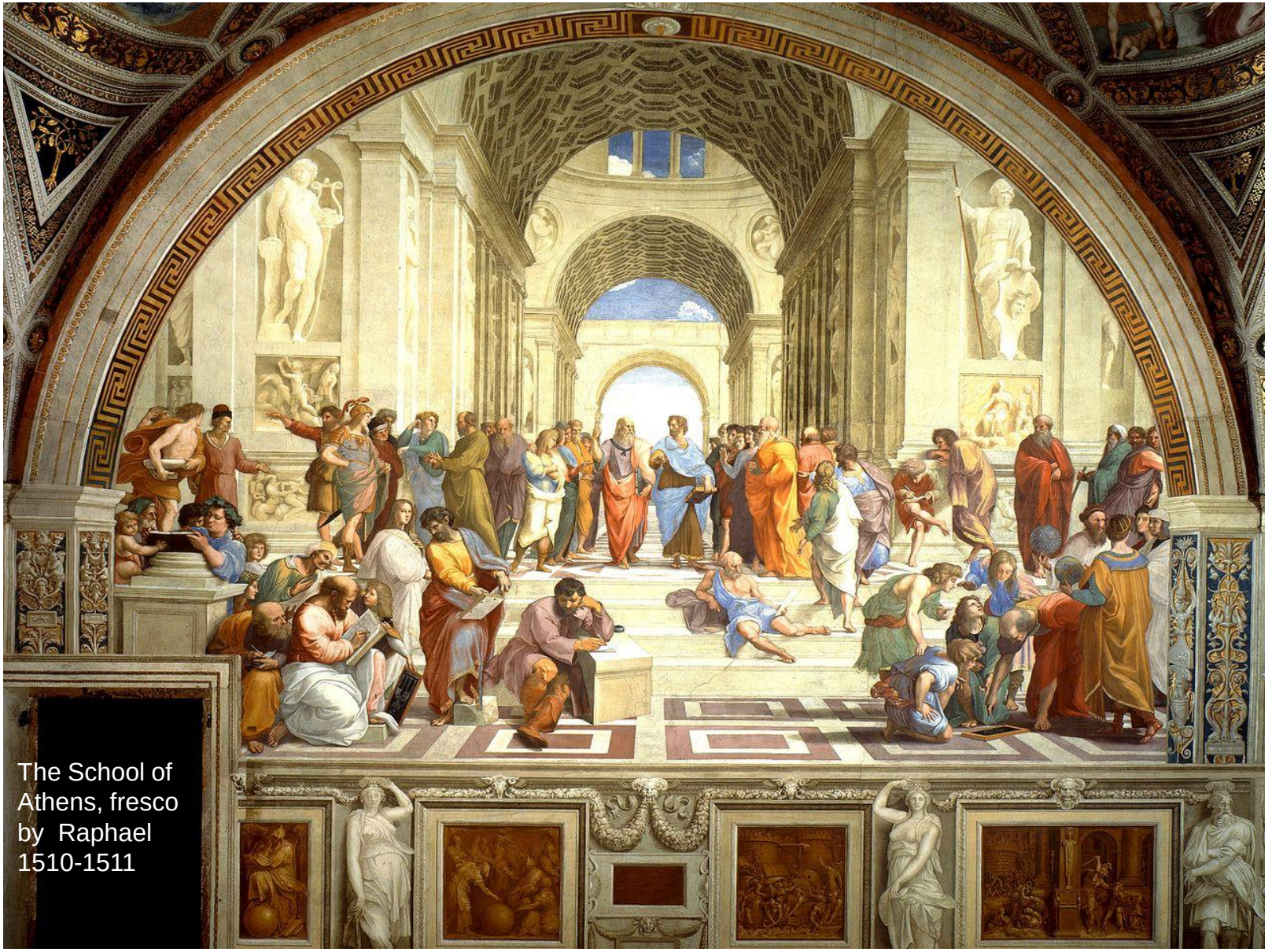
In: S.B. Cooper & JvL (Eds) Alan Turing - His Work and Impact, Elsevier, 2013



- *Why* do we compute?
- *What* do we compute?

?

2. Philosophy of… Computing



The School of Athens, fresco by Raphael 1510-1511

Great issues

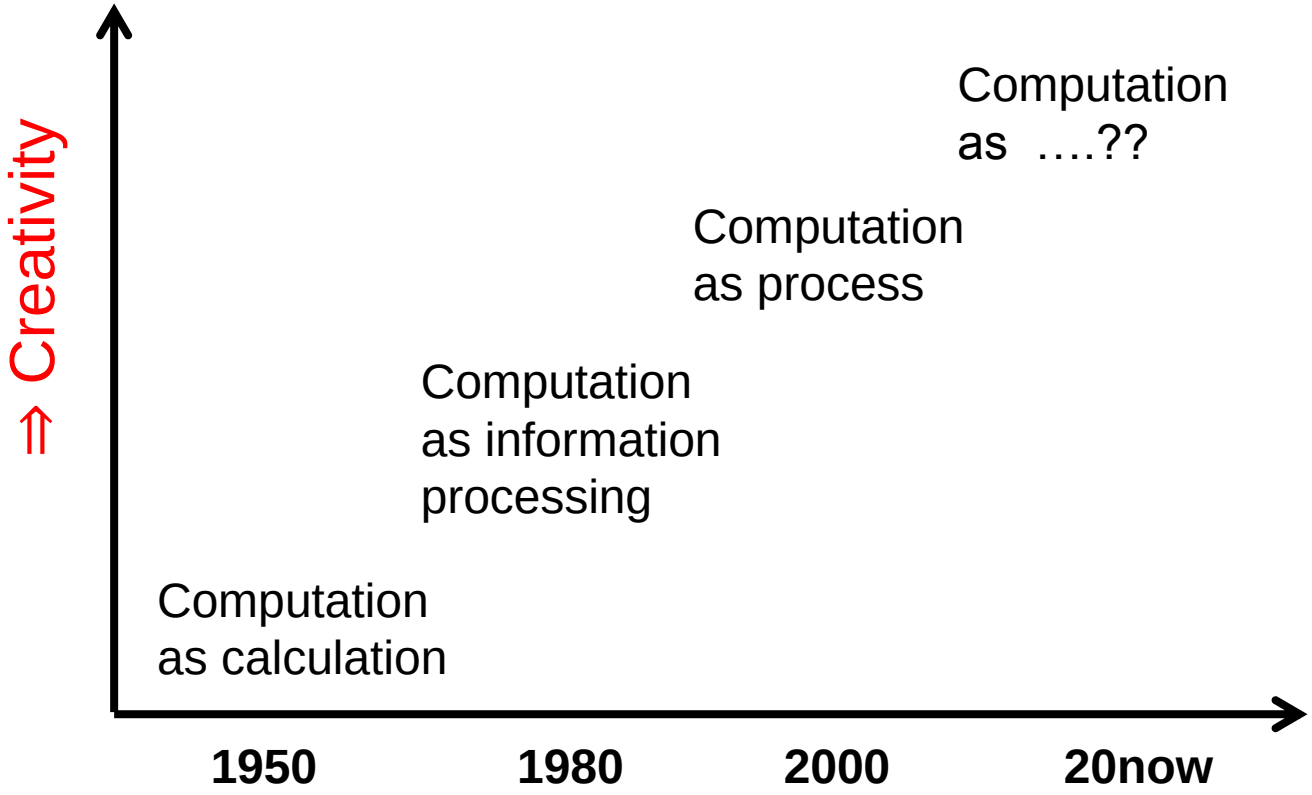
- **Object(s) of study**
  - Natural science, engineering, design, back-casting, …
- **Development**
  - Historical lines, paradigms, is science or technology leading?
  - Centric views: logic, mathematics, programming, engineering, specific applications, etc.
  - Influences: scientific, technological, social, societal, political, …
  - `From tool perspective to science perspective', what leads the transformation?
- **Philosophy of `perspectives'**
  - Impact of centric views
  - Information-oriented, computing-oriented, communication-oriented, cognition-oriented, design-oriented, behavior-oriented,
  - Sub-areas (AI, agents, …)
  - Concepts in context
- **Concepts and methods**
  - Ontology, epistemology, what is understanding in Informatics.
  - Fundamental notions: how to understand computation, information, programs (formally, o/w?)
  - Abstraction, stepwise modeling, algorithms & mechanisms, languages, specification, design methods
  - Methodology: theory, experiment, *program (`third way')?* Productize*?*
- **Impact**
  - **On our thinking**
    - Algorithmic (computational) thinking, design science, social computing
    - Awareness of resources, complexity, `think parallel' (Intel), …
    - Reviving "can machines think, have emotions, etc" (mimicking behavior by mass computation)
  - **On our values**
  - **On our future**

Computation is ??

- **Early conceptions**
  - R. Lull (1230-1315): “mechanically generate all possible thoughts or ideas…”
  - Th. Hobbes (1588-1679) : “all thought is a form of computation.”
  - G.W. Leibniz (1646-1716): “reduce all reasoning to calculation.”
- **Evolution**
  - Calculation ⇒ Mathematical method to determine something effectively ⇒ discrete and scientific computing ⇒ …
  - Record keeping ⇒ business data processing ⇒ information technology ⇒ data & knowledge engineering ⇒ …
  - Automating ⇒ …
- **Now: Shift towards know-ledge-enriched processes**
  - Making all systems ‘intelligent’
  - Self-adjusting (self-modifying, interactive, evolving)
  - Cognitive abilities (learning, creativity, consciousness, …)
- ***Anything that can be (or we want to be) understood as computation***
  - Everything virtual: simulation -> visualization -> interaction -> experience -> emotion -> …
  - The *Algorithmic Lens* (Karp) , explanatory power
  - New way of doing/understanding science (Kuhn)?
  - … of *discovering knowledge*?

3. Our view of computation

What is the question?



⇒ Understanding of computation

(Some) recent opinions:

L.G. Valiant: “computation [is] the execution of step-by-step procedures for processing information.”

J. Searle: “Computation does not name a machine process. It names an abstract mathematical process that we have found ways to implement in machines.”

Y. Gurevich: “computation is the evolution of states.”

S. Akl: “Computation is the process (or a collection of processes) of acquiring information, transforming it, and providing the outcome to the outside world. At the very bottom everything is a computation.”

P.S. Rosenbloom : “Computation is information transformation.”

D. Frailey: “Computation in its broadest sense is anything that happens (as opposed to things that are static). If so, then the principles of computation are, In fact, the principles of processes.

D. Deutsch: “A computation is a physical process in which physical objects like computers, or slide rules or brains are used to discover, or to demonstrate or to harness properties of abstract objects — like numbers and equations.”

The current views:

- generally tend strongly towards observer-*in*dependence,
- view computation as a mechanistic process,
- *are defined by what an underlying machine (or model) is doing*, i.e. *by how the process is realized,*
- do not cover many cases adequately.

But:

All our experience with computation points in another direction: we are primarily interested in ***WHAT* the computation ultimately does** for us **!**

Thesis:
Computation is a (any) process of *knowledge generation* (in a suitable knowledge space).

J. Wiedermann & JvL, `Rethinking computation', in: 6th AISB Symp. on Computing and Philosophy, Exeter, 2013.

Computation…

Classical approach:
How do we compute

Many philosophical analyses after Turing

Computation

- …as a process
- …as information transformation
- …as symbol manipulation
- …as any process described by some formal model of computation (e.g. a Turing machine)
- …that that must have a physical realization.

Observer-independent.
Intrinsic to physics.

Epistemic approach:
What do we compute

JW & JvL 2013

***Computation is a process of* knowledge generation**

Observer-relative.
Intrinsic to the relevant epistemic theory.

What is… knowledge?

Knowledge is a familiarity with someone or something, which can include facts, information, descriptions, or *skills* or *behavior* acquired through experience, education or copying.

It can refer to the theoretical or practical understanding of a subject. It can be implicit (as with practical skill or expertise) or explicit (as with the theoretical understanding of a subject).

It can be *more* or *less* formal or systematic.

Plato, Aristotle, …

Knowledge as justified, true, belief.

Examples: Computation as knowledge generation

CONTEMPORARY COMPUTING SYSTEMS		
Computational system	**Underlying knowledge domain**	**What knowledge is produced**
Acceptors	Formal lang. & Automata	Acceptance
Recognizers	Formal lang. & Automata	Membership function
Translators	Functions, relations	Function values
Scientific computing	Mathematics	Solutions
Theorem provers	Logic	Proofs
Database and information systems	Relations over structured finite domains	Answers to formalized queries
Search engines	Relations over unstructured potentially unbounded domains	Answers to queries in a natural language
Artificial cognitive systems	Real world , science	Conjectures, explanations

NATURAL COMPUTATION

Computational system	Underlying knowledge domain	What knowledge is produced
Living systems, cells	Real world, biology	Life, behavior, intelligence
Brain, mind, social systems	Knowable world	Knowledge of the world
The Universe	The Universe, physics, life	Life, materialized knowledge

NON-TURING COMPUTATIONS

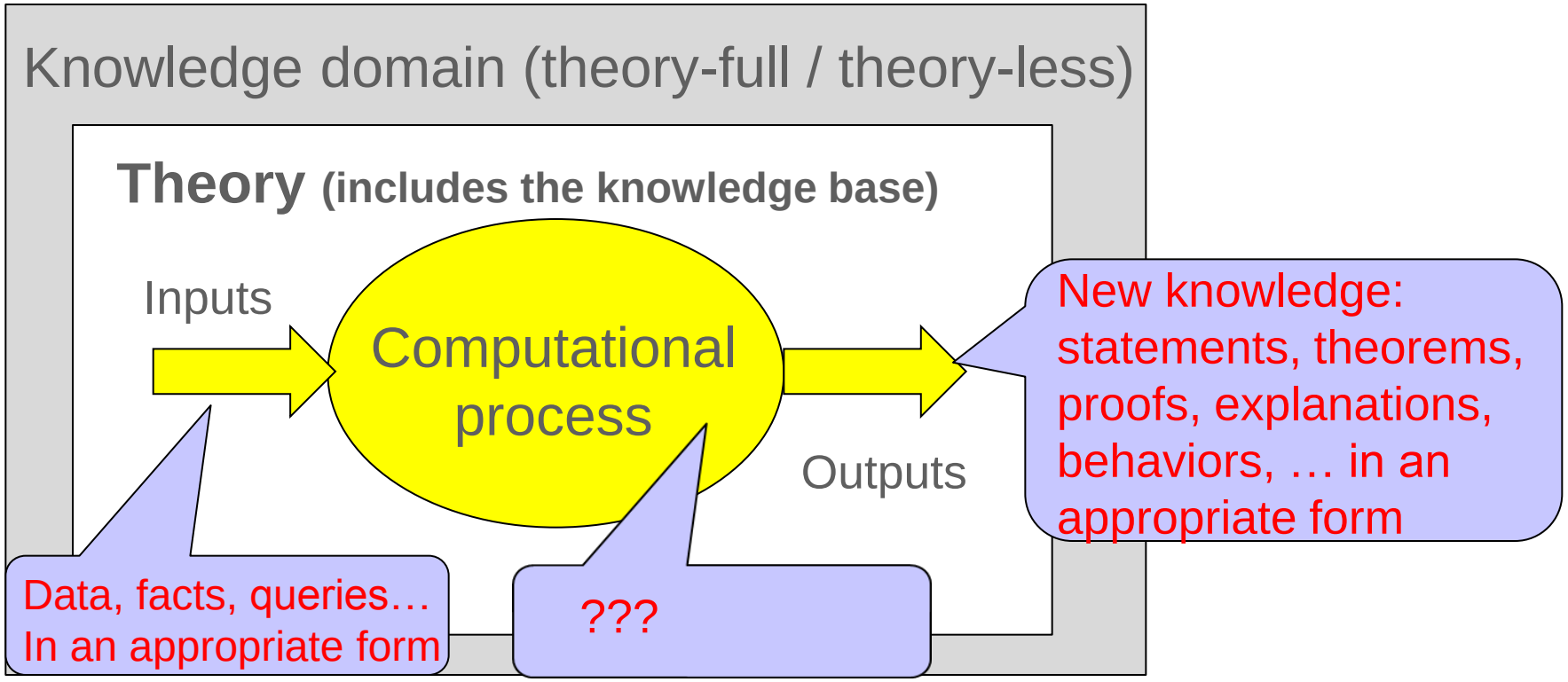
Computational system	Underlying knowledge domain	What knowledge is produced
Compass and ruler	Euclidean geometry	Euclidean constructions
BSS computer	Real numbers	Values of rational real functions
Oracles	Subsets of natural numbers	Characteristic function of those subsets

Varieties of Knowledge

	Theory- full -------- KNOWLEDGE DOMAIN **--------Theory-less**		
	Logic and mathematics	***Natural sciences and philosophy***	***Mind and humanoid cognitive systems***
Domain of discourse	Abstract entities	Empirical data, ideas	Perception
Initial elements of knowledge	Axioms, definitions	Observations, facts	Beliefs, stimuli, multi-modal concepts, episodic memories
Inference rules	Deductive system	Rational thoughts, logic	Rules and associations emerging via learning
Final form of knowledge	Sentences, theorems, proofs	Statements, theorems, hypotheses, explanations, scientific laws, predictions	Conceptualization, behavior, communication, natural language thinking, world knowledge expressed in a natural language and in form of theories

4. Computation as an (observer-relative) notion

There is an epistemic “theory” behind each computation



Natural language as a mediating tool among heterogeneous knowledge

Knowledge domains E covered by the theory ideally are *deductively closed: if Φ deducible from a (finite) subset of E, then Φ ϵ E*.

Defining computation

- T - a theory;
- ω - a piece of knowledge serving as the input to a computation;
- κ - a piece of knowledge from T denoting the output of a computation;
- Π - a computational process, and
- E - an explanation;

Definition: Process Π, acting on input ω, generates piece of knowledge κ if and only if the following conditions hold:

- (T, ω) ├ κ, i.e., κ is '*derivable*' within T from ω, and
- E is the (causal) *explanatio*n that Π generates κ on input ω.

We say that C = (T, ω, κ, Π, E) is a computation rooted in theory T.

The physical or abstract *entity realizing process* Π is called the underlying mechanism (or computer).

Computation as an observer relative process

Definition: an *observer* is a computation O which decides, upon observing another process C = (T, ω, κ, Π, E), whether C is a computational process.

Ideally, C = (T, ω, κ, Π, E) serves as an input to O. In many cases (especially in theory-less domains), only pairs <ω,κ> are available to O; the remaining information must be "discovered" by the observer.

O must check whether

(a) κ follows from T and ω, and

(b) E is the explanation that Π generates κ on input ω.

A decision of O that C is a *computational* process depends on the (computational) ability of O to verify conditions (a) and (b).

Proposition. There exists no universal observer whose verdict always agrees with the verdict of every other observer, for each 5-tuple C = (T, ω, κ, Π, E).

43

5. A theoretical model (in progress)

Capturing computations

- Metaspaces
  - *Action space* (of the underlying mechanism)
    - Action items
      - Examples: observable cell descriptions, full information descrip-tions of ASM, or of computer.
    - Computation respects a proximity relation.
    - ***Postulate***: action spaces are topological.
    - ***Postulate***: represented action spaces are metrical.
  - *Knowledge space* (of knowledge items we know or aspire to know and/or produce)
    - Framework, formal or informal
      - Examples: first-order 'worlds' describing the `knowledge' of a state reached in an ASM.
- Common features
  - Core set of accessible/known items.
  - Metaspace discovery.

What is a computation?

- Action space A, knowledge space E
  - Cores A_0 and E_0
- Semantic map δ: $\mathrm{A} \rightarrowtriangle \mathrm{E}$
  - Partial map with $\delta(A_0) \subseteq E_0$, no additional computation.
- ***Definition***: a computation is any *curve* c in A such that
  - $\delta(c^{init})$ is defined,
  - If c^{end} is defined, then $\delta(c^{end})$ is defined as well.
- ***Definition***: a bundle is any collection of computations $B = \{c_i\}_{i \in I}$.
  - Examples: behaviours of cells, nervous system, internet search.
- Notions:
  - Computation c is *enabled* once $\delta(c^{init})$ is known.
  - If c enabled and convergent, then $\delta(c^{end})$ is *computed knowledge item*… but *also* rest of $\delta(c)$.
  - Composition of computations $(c \circ d)$, grafted composition $(c \Delta d)$.
- Cross connections: control theory, trace theory, computable topology.

Exploring spaces by computation

- Knowledge *generation*
  - with bundle $B = \{c_i\}_{i \in I}$.
  - $e \in E$ is producible by $c_1, \ldots, c_k$ from B (denoted $c_1, \ldots, c_k \vdash e$) if ${c_1}^{init} \in E_0$, and $e \in \delta(c)$ for some $c \in c_1 \Delta \ldots \Delta c_k$.
- K_B : all knowledge items that can become known using bundle B.
  - ***Proposition***: K_B is the least fixpoint of an operator G defined over 2^E (hence well-defined). If B is compositional, then $K_B = G(E_0)$.
- Notion:
  - A bundle B is universal for E if and only K_B = E.
- *Knowledge recognition*
  - Counterpart to generation.
  - ***Proposition***. Given bundle B, a computational process can be designed which precisely `recognizes' all items from K_B , based on the computational mechanism underlying B.

When is a process computational?

- Underlying mechanism is intuitively a process of knowledge generation (observer).
- Specify spaces
  - Action space, knowledge space, core sets.
  - Semantic map.
  - Bundle of computations (`curves').
- Need not specify how it works, but justify the knowledge generation
  - Explanation in suitable framework.
  - Representation (Fodor)
    - Observer-dependent.
    - Represented action spaces (e.g. ASM).
- Determine that the underlying mechanism is a process of knowledge generation against the backdrop of a suitable domain theory (wrt observer).

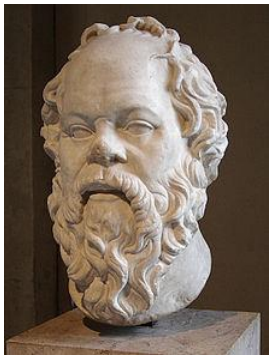
Examples: cell, ASM,…

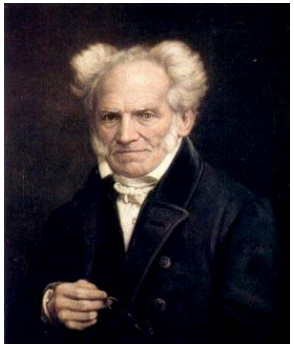
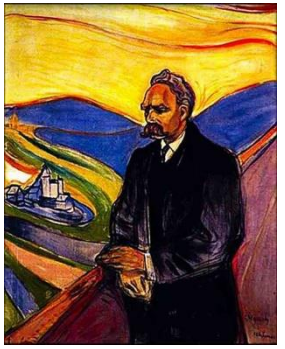
Computation as knowledge generation

- Separates computing and non-computing objects
  - *relative to an observer*.
- Allows classification of computations
  - *wrt quality and quantity of produced knowledge*.
- Not depending on the underlying operational model of a process
  - *Covers both known and yet unknown instances of computation.*
- Allows considering computations at a high abstract level.
  - *`how to compute it', test for algorithmic mechanisms*
  - *the central dogma resp. ASMs are instances*
  - *new theory of computation*
- Resolves notorious problems in the scope of the classical definition.
  - *According to new definition*, cognition is computation.
  - *Ecorithms are computations*
- Focuses on the intrinsic meaning of computations
  - *Consequences in AI, philosophy, epistemology, methodology of science, cognitive sciences, …*
  - *Epistemic question: identify knowledge generation*
- Resolves Abramsky's questions!
  - *What do we compute, why.*
- New meaning to the computation-centric perspective
  - *Understanding science is explaining computations?*.

6. Towards a computational theory of creativity

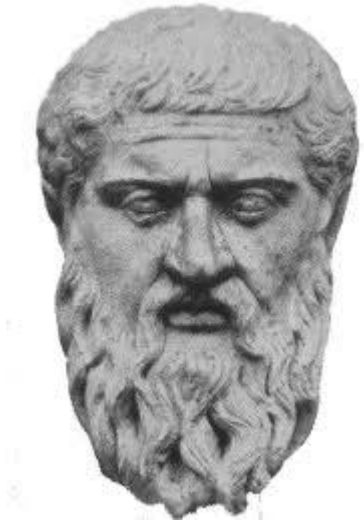
What is creativity?



- Socrates (470-399 BC) : divine inspiration, by Muses
- Kant (1724-1804): innate capacity of artistic genius
- Schopenhauer (1788-1860): capacity of the greatest artist s to “lose themselves” in the experience what is beautiful and sublime
- Nietzsche (1844 – 1900): capacity born out of rare cooperation between ecstatic intoxication and sober restraint

Greek philosophers like Plato rejected the concept of creativity, preferring to see art as a form of discovery. Asked in The Republic, "Will we say, of a painter, that he makes something?", Plato answers, "Certainly not, he merely imitates."



20th century: “The creative act is not an act of creation in the sense of the Old Testament. It does not create something out of nothing: it uncovers, selects, re-shuffles, combines, synthesizes already existing facts, idea, faculties, skills.” (Koestler, 1964)

Wikipedia: *Creativity* is a phenomenon whereby something new and in some way valuable is created.

Our stance:

1. Computation is knowledge generation (previous part of this talk).

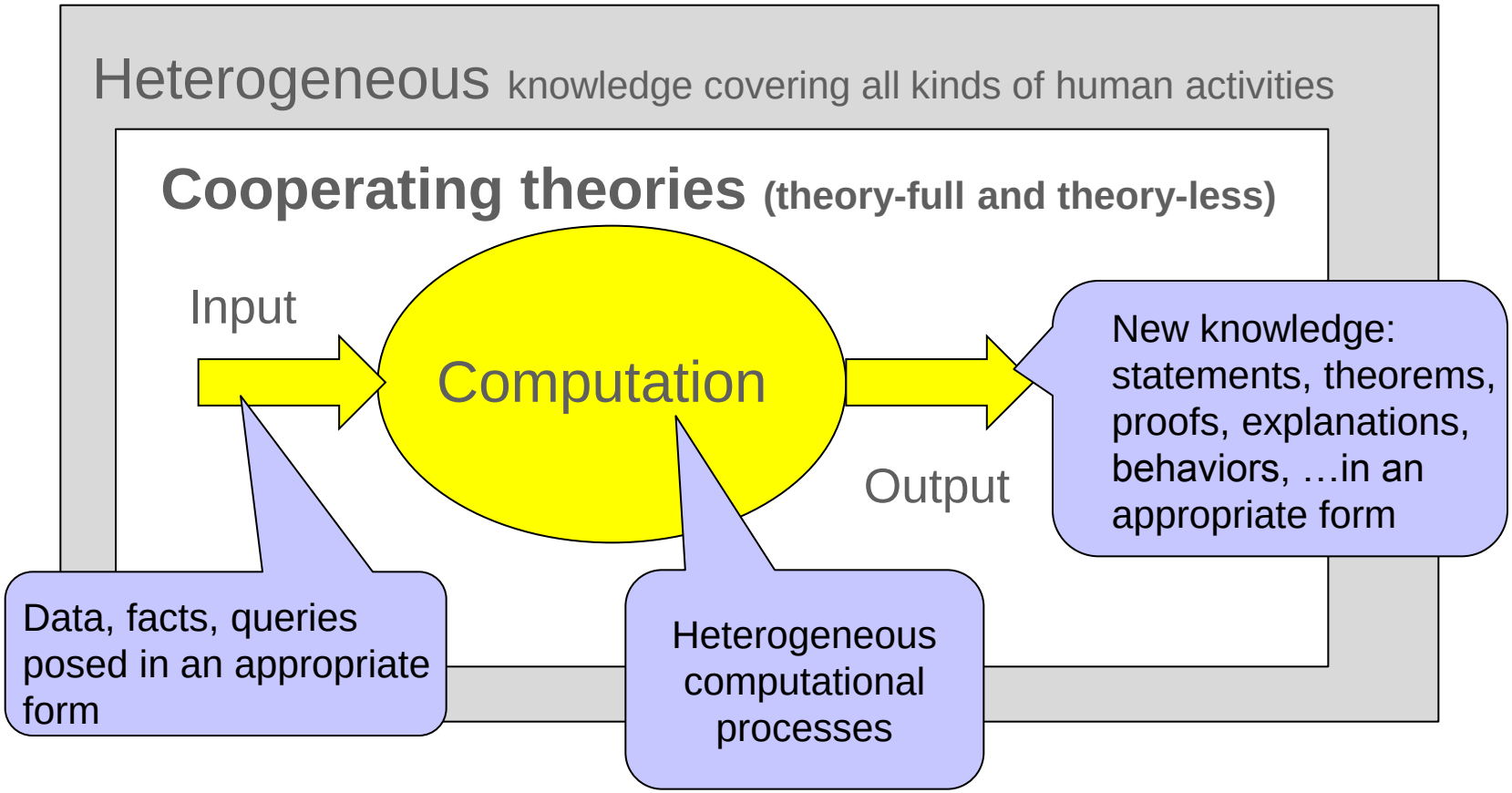
2. **Creativity is knowledge generation in the form of ideas, artifacts or behavior whereby something new and in some way valuable is produced.**

Questions to be answered:

How does creative knowledge production fit into our framework?

When can a computational process be seen as a creative process?

How does creative knowledge production fit into our framework?



An “appropriate form” means a form adequate to the domain of discourse.

What's wrong with the current epistemological approach to knowledge creation:

Known epistemological processes describe knowledge generation as extrapolations of repeated observations or known facts, usually as variants of inductive reasoning.

Ecclesiastes (Kohelet):
(3rd century BC)

ט. מַה שֶּׁהָיָה הוּא שֶׁיִּהְיֶה וּמַה שֶּׁנַּעֲשָׂה הוּא כָּל חָדָשׁ תַּחַת הַשָּׁמֶשׁ: שֶׁיֵּעָשֶׂה וְאֵין

"What has been is what will be,
and what has been done is what will be done,
and there is nothing new under the sun"

What is needed:

The ability to generate genuinely new knowledge, in a creative way.

The ability to create new explanations, not mere extrapolation or generalization of the past experience.

What is analogy

- basic act of knowledge generation requiring an explanation

- analogy is reasoning or explaining from parallel cases, or

- analogy is a figure of language that expresses a set of like relations between two sets of terms

Explanatory analogy: creates understanding between something unknown by relating it to something known.

It provides insight or understanding by relating what one does not know with what one knows.

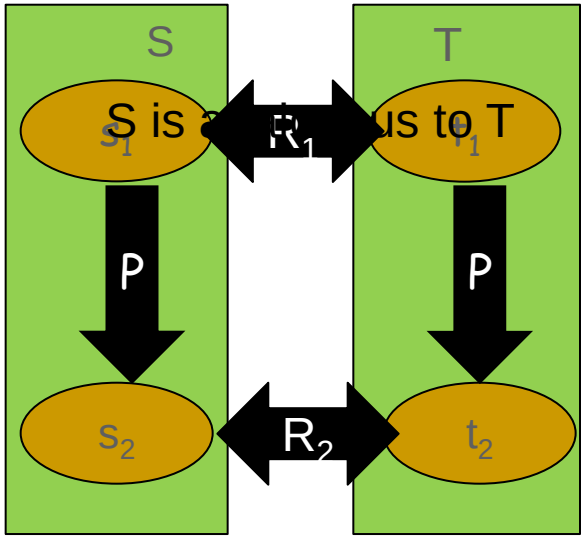
Analogies refer to natural language, understanding, reasoning, explanation, and creativity

Our goal:

Approaching analogies from the viewpoint of knowledge generation

Defining analogy

Definition:
Let $S=(s_1,...,s_k)$ and $T=(t_1,..., t_k)$ be two sequences of linguistic expressions.
If there exists a linguistic k-ary predicate P such that both P(S) and P(T) hold and linguistic relations $R_1,\ldots,R_k$ such that $R_i(s_i,t_i)$, for i=1,… k holds, then we say that **S is analogous to T** w.r.t. predicate P and relations $R_1,\ldots,R_k$.



Given S and T, **analogy** is a cognitive process whose purpose is to find linguistic predicate P and linguistic relations $R_1,\ldots,R_k$ such that S is analogous to T w.r.t. predicate P and relations $R_1,\ldots,R_k$.

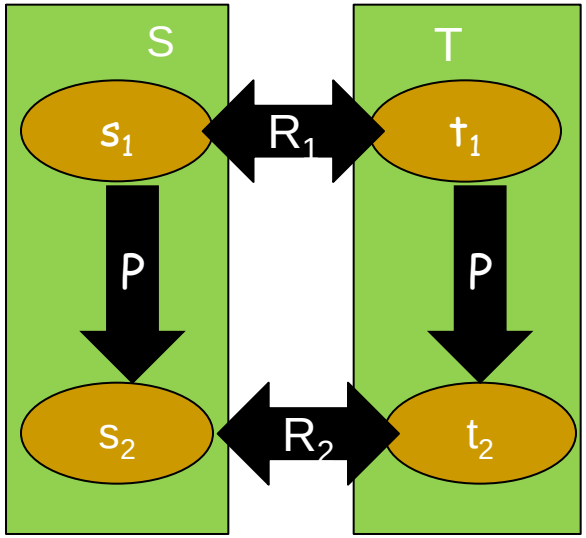
We say that P is a **conjecture** and P(S), P(T) and $R_1,\ldots,R_k$ are the **explanation** of this conjecture.

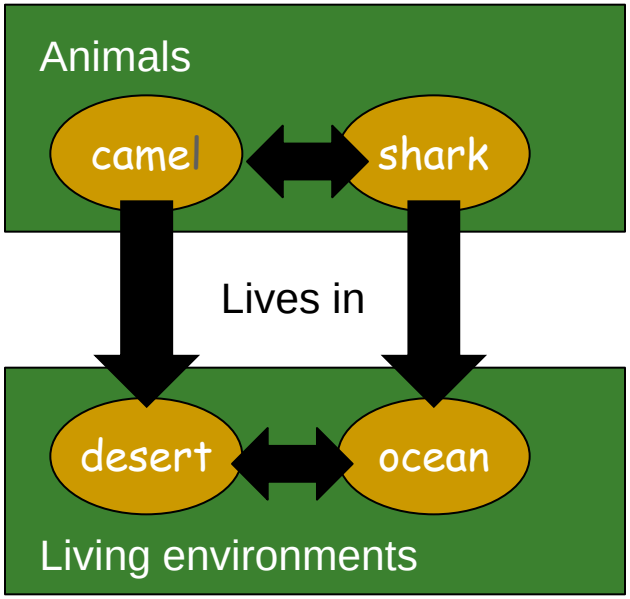
An example of an analogy:

shark : ocean :: camel : desert

Conjecture P: "x lives in y"

Explanation
P(S): "camel lives in a desert""
P(T): "shark lives in ocean"
R_1: "both shark and camel are animals"
R_2: "both ocean and desert are living environments"

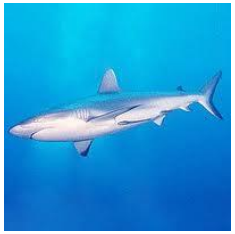




Explanatory analogy:
find **x** such that

shark : ocean :: **x** : desert

Variants of analogies:
- incomplete analogies
- metaphors
- allegories



Analogy can also be resolved from pictures

Solving incomplete analogies

Input: k-ary predicates S and T with some attributes unspecified; knowledge base K
In order to resolve such analogy, we need to discover the following knowledge:

• We have to check whether an object corresponding to predicate S does exist in K. If not, the answer would be ``I don't know" and we are done.

• For each object T' from K satisfying predicate T in specified attributes, we check whether in K there exists:

- a k-ary predicate P satisfying P(S)=P(T') (i.e., we are looking for a conjecture). If there is no such P the answer would again be ``I don't know" and we are done.

- next, we look for linguistic relations of similarity $R_1,\ldots,R_k$ in K such that R_i (s_i ,t_i) for i=1,…,k holds. If such relations are found then the output returns object T', conjecture P and explanations R_i's. Otherwise the answer would again be ``I don't know" and in either case, we are done.

If no object T' is found then the answer is ``I don't know" and we are done.

Instead of "I don't know" answers we can look for the missing knowledge in "external sources" like the Internet, encyclopedias, monographs, experts…

Rather than created the solution of an analogy is discovered.

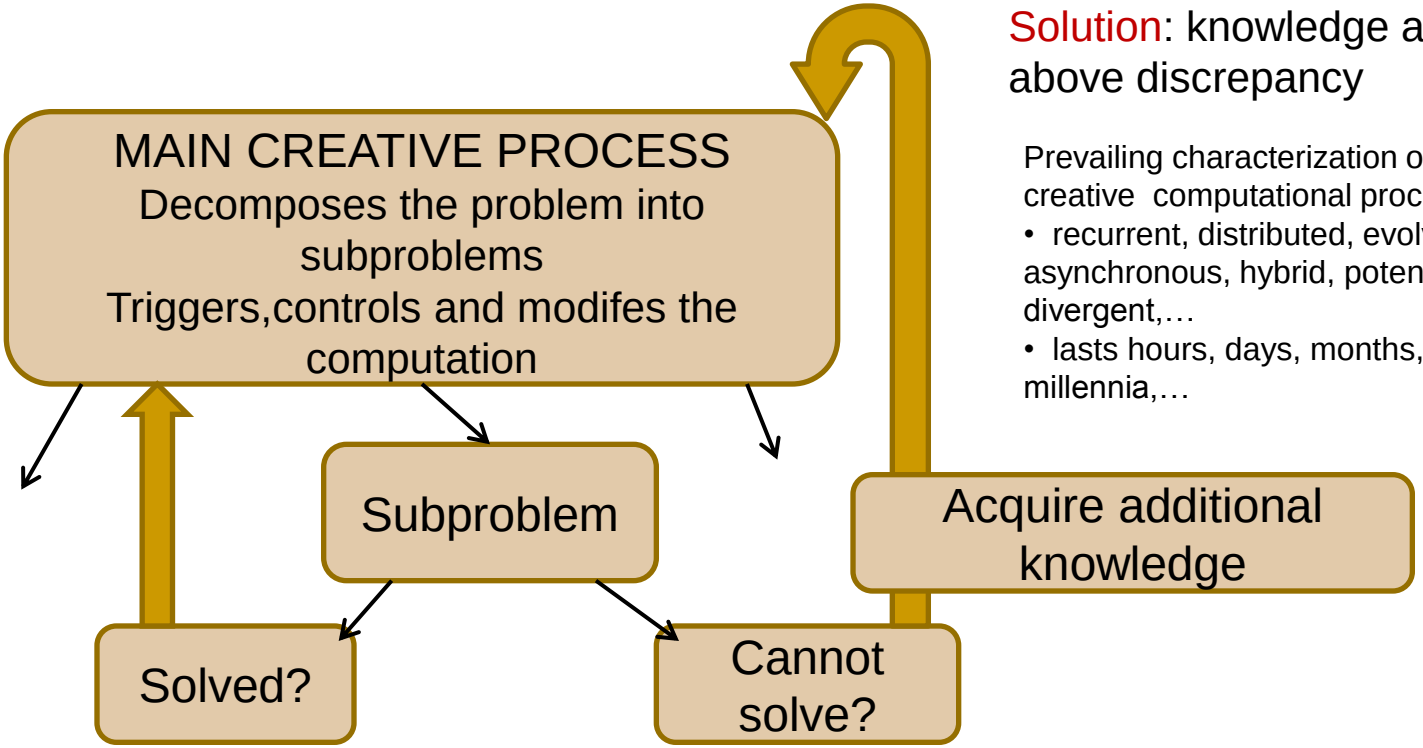
When can a computational process be seen as a creative process?

A creative process is such that solves a new problem.

A routine process solves a known problem.

Problem: a discrepancy between prediction (generated knowledge) and the observation (observed knowledge)

Solution: knowledge avoiding the above discrepancy



Prevailing characterization of a creative computational process:

- recurrent, distributed, evolving, interactive, asynchronous, hybrid, potentially infinite, divergent,…
- lasts hours, days, months, years, centuries, millennia,…

Message to take home

Knowledge is not created by some divine inspiration, rather it is being discovered within the existing knowledge domain and added to it.

Creative processes generate knowledge solving new problems, i.e. problems whose solution has not been known to the knowledge creator.

Knowledge discovery process is an endless process rejecting old knowledge and accumulating increasingly more new knowledge

7. Conclusion

Computation is knowledge generation

"Creation of knowledge ... now has to be understood as one of the fundamental processes in nature; fundamental in the sense that one needs to understand them in order to understand the universe in a fundamental way."

D. Deutsch

"Computation ... now has to be understood as one of the fundamental processes in nature; fundamental in the sense that one needs to understand them in order to understand the universe in a fundamental way."

JvL & JW

The more we understand the Universe the more computation (hence knowledge) we see around us.

DISCLAIMER

These slides are for use at SOFSEM 2015 only. They are not meant for distribution or posting in public media outside the realm of the conference.

All materials are intended for educational use at SOFSEM only. Illustrations, tables and other possibly copyrighted materials should be cited by their original sources, including our own.

Any violations of copyright will be removed once they become known to us.