Programs as tools for knowledge
Henri Salha, IHPST – Paris-I University

HAPOP4 symposium
Oxford, 23rd March, 2018
What can we learn from computer programs?
Some preliminary definitions

Scope restriction: programs at run-time, i.e. computational processes

A computational process is either (1) the execution of a rule-based sequence of pre-defined operations, by a digital computer, involving the handling of data or exchange of data with an external context; or (2) the rule-based parallel execution of many computational processes.

What is the context of a process?
Why this question? Cognitive sciences, computational science

Dennett 1980:

“AI [shares] with traditional epistemology […] the top-down question: how is knowledge possible? […] It shares with psychology in distinction from philosophy a typical tactic [which is to] ask themselves an easier preliminary question:

“How could any system (with features A, B, C, …) possibly accomplish X?”

This is an engineering question, a quest for a solution (any solution) rather than a discovery”

Humphreys 2004:

Computational methods now play a central role in the development of many physical and life sciences. […] […] [These] developments, which began in the 1940s and accelerated rapidly in the last two decades of the 20th century, have given rise to a new kind of scientific method that I shall call computational science. (p49)
Computer simulations (1)

A core topic of epistemology, with well known problems

- Empirical vs. a priori: status of results as theoretical extrapolations or genuine observations
- Justification ("Simulations are only as good as their assumptions")
- Epistemic opacity: practical inability of the epistemic agent to check the computations

... But also an issue to define which programs should count as simulations

- How to distinguish simulations from simple numerical calculations programs?

Humphreys 2004, p112

In the early days of computer simulations, static numerical arrays were all that was available, and it would seem unreasonable to disallow these pioneering efforts as simulations. [...].

The definitions we have given up to this point might seem to give us no reason to claim that computer simulations are essentially different from methods of numerical mathematics.
« A simulation imitates one process by another process. In this definition, the term “process” refers solely to some object or system whose state changes in time. »

1) Imitation

2) Time
About the time dimension of programs: two broad families

Manna & Pnueli, 1992, p.3

“A transformational program is the more conventional type of program, whose role is to produce a final result at the end of a terminating computation. Consequently, the useful view of a transformational program is to consider it as a […] function from an initial state to a final state or a final result. […] For such specifications, ordinary predicate logic provides an adequate formulation and reasoning tool.

The role of a reactive program, on the other hand, is not to produce a final result but to maintain some ongoing interaction with its environment. Examples of reactive programs are operating systems and programs controlling mechanical or chemical processes, such as a plane or a nuclear reactor. […] They cannot be specified by a relation between initial and final states, […] It is for] such programs that the formalism of temporal logic […] is recommended. ”

See also Lehman 1980, distinction between S & E programs is somewhat similar
Epistemic power of functional programs

Data analysis
(aka Knowledge Discovery from Databases)

- Archiving & communication
- Data organization & visualization
- Finding, grouping & sorting
- Classification & Regression
- Data description (eg clustering)
- Pattern recognition
- ...

Analytical developments

- Translations & transcoding
- Maths analysis
- Combinations & developments
  (eg tree exploration)
- Iterations and recursion
- ...

Epistemic power of reactive systems

Knowledge in action – “know-how”, ability

But only indirect knowledge for the observer
  • Data extraction for functional analysis
  • Performance analysis, learning from experiment

No need of formal semantics

Brooks, 1991:
Even at a local level we do not have traditional AI representations. We never use tokens which have any semantics that can be attached to them. The best that can be said in our implementation is that one number is passed from a process to another. [...] To a large extent the state of the world determines the action [and not a model of the world]
Two interpretations of knowledge map broadly with our two families of computations

<table>
<thead>
<tr>
<th>“Know-that”</th>
<th>“Know-how”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbolic knowledge</td>
<td>Practical knowledge</td>
</tr>
<tr>
<td><strong>Output converts to…</strong></td>
<td><strong>Representation</strong></td>
</tr>
<tr>
<td><strong>Output values to…</strong></td>
<td><strong>Truth</strong></td>
</tr>
<tr>
<td><strong>Computation means…</strong></td>
<td><strong>Relationships (laws)</strong></td>
</tr>
<tr>
<td><strong>Context</strong></td>
<td><strong>Abstracted (black-boxing)</strong></td>
</tr>
<tr>
<td><strong>Epistemic warrant</strong></td>
<td><strong>Semantic model</strong></td>
</tr>
<tr>
<td><strong>Thought Traditions</strong></td>
<td><strong>Functional cognitivists</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Philosophers of Mind</strong></td>
</tr>
</tbody>
</table>
The scale of concurrency and openness: any intermediate cases?

Pure functional programs

- Sequential
- No context\(^{(1)}\)

Pure reactive systems

- Concurrent
- Open context

\(^{(1)}\) Except at start-up, as inputs
Intermediate cases?

Testing / training reactive systems in controlled environments
Aiming to shut down any unexpected event
Other examples: tests of new software through mass data feeds

Testing and training users in controlled environments
Part of the context is data fed (e.g. the travel data), part of the context is object of the test (e.g. the apprentice-pilot)

Games are the general form of « closed context » processes
Concurrency is still very much alive, but the computational process is embedded in a context which is ideally shut down from reality: gamers « play a role » with pre-defined moves
Further cases

The logic can continue and a given context can always be modelled to be embedded in a larger process

Illustration: Neuronal Network trained to play and win Super-Mario

• The gamer is emulated by another computational process interacting with the game process
• Together they form a new concurrent process, with no external active context

« Zero-player games » are pure cases which are both functional (no context) and concurrent – all the context has been virtualized

• Illustration: Robotwar, first programming game where the players’ interventions take place before the actual battle begins
• Game of life, one of the first agent-based simulations

Other multi-variable numerical iterations, such as Euler method for simulating the 3-body problem, taking the form of pseudo-concurrency (round-based updates of the bodies cinematics) may be interpreted as « games » in this latest sense
The scale of concurrency and openness

Pure functional programs
Simple variable iterative computation

Agent-based models
Interdependent variables iterative computations

Video-game + Emulator
Agent-based models

Tests of reactive systems
Video-Games Simulators

Pure reactive systems

- Sequential
- No context

- Pseudo-concurrent
- No context

- Concurrent
- No context

- Concurrent
- Closed context

- Concurrent
- Open context

« Simulations »

« Games »
Conclusions

Three categories of computer programs which show different uses in knowledge endeavors

• Functional programs covering analytical functions (incl. induction)
• Reactive programs covering technical and practical functions
• Games at the interplay between these two sides of knowledge

Could this categorization prove helpful to the debate between functional cognitivists and dynamical / situated cognitivists?

Other leads for exploration:

• Epistemological issues linked to simulations
• Ontology of computational processes?
• Epistemology of programming
Barberousse, Anouk, Sara Franceschelli, and Cyrille Imbert, ‘Computer Simulations as Experiments’, Synthese, 169 (2009), 557–74
Dowek, Gilles, Les Métamorphoses du calcul (Humensis, 2015)


Hughes, Richard IG, ‘Models and Representation’, *Philosophy of Science*, 64 (1997), S325–S336


———, ‘Understanding Programming Languages’, *Minds and Machines*, 17 (2007), 203–16


