Semantics for Physicists

Prakash Panangaden[1]

[1]School of Computer Science
McGill University

Workshop on Compositionality 5th December 2016

Our shared concerns

Describe the evolution of a *precisely specified* dynamical system.

Programming Semantics	Theoretical Physics
Precise syntax	Formalism precise but implicit
Execution effect: operational semantics	Hamiltonian/Lagrangian: differential equations
Mathematical model: denotational semantics	What goes here?: phase portrait??
Compositional description	Compositionality becoming important
Types	Types are emerging

Benefits of semantics

- Denotational semantics gave a *compositional* account of program behaviour.
- Paying attention to the formal semantics leads to new ideas and new constructs: higher types, recursive types, continuations, coinduction, monads.
- A compositional theory allows one to reason about *open systems*: e.g. verification of open systems via game semantics.
- Category theory provided a powerful organizing framework for compositional thinking.

Compositionality in physics

- Tensor product as the way to combine quantum systems.
- Coupling of fields through “interaction terms”.
- Feynman diagrams: a diagrammatic way of keeping track of perturbation series.
- Categories emerging as a way to think about composing physical system.
- Categorical quantum mechanics: Abramsky, Coecke, Selinger, Kissinger, ...
- Baez, Eberle, Fong, Pollard...: categorical view of Markov processes, circuits, linear systems, Feynman diagrams, networks

An outline of semantics

- Programs are viewed as inductively defined terms, e.g.
- Syntax of commands

$$\mathsf{com} \quad ::== X \longrightarrow \mathsf{exp} \;\mid\; \mathsf{com}_1;\mathsf{com}_2 \;\mid\; \textbf{if}\ \mathsf{bexp}\ \textbf{then}\ \mathsf{com}_1\ \textbf{else}\ \mathsf{com}_2$$

$$\mid\ \textbf{while}\ \mathsf{bexp}\ \textbf{do}\ \mathsf{com}\ \textbf{od}$$

- A *state* is a map from variables to values.
- A command is *interpreted* as a *partial* function from states to states.
- $[\![com]\!]$: State $\rightharpoonup$ State
- $[\![\cdot]\!]$ can be defined compositionally, e.g.
- $[\![c_1;c_2]\!] = [\![c_2]\!] \circ [\![c_1]\!]$.
- How about **while**? Even in this very simple language one needs to use fixed-point theory.

Operational semantics and denotational semantics

- Operational semantics : explicit step-by-step description of execution as state changes.
- Can be made *close to* compositional but
- iteration, recursion (and some other things) cannot be given strictly compositionally.
- Denotational semantics uses a mathematical model of programs as functions but order and topology are essential ingredients.
- Using fixed-point theory one can give a completely compositional semantics.
- Why should one believe that a mathematical model actually corresponds to what happens when one executes a program?
- Someone has to prove correspondence theorems between these two types of semantics: adequacy, full abstraction.