Semantics for Physicists

Prakash Panangaden\textsuperscript{1}

\textsuperscript{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**

- Precise syntax
- Execution effect: operational semantics
- Mathematical model: denotational semantics
- Compositional description
- Types

**Theoretical Physics**

- Formalism precise but implicit
- Hamiltonian/Lagrangian: differential equations
- What goes here?: phase portrait??
- Compositionality becoming important
- Types are emerging

Panangaden (McGill University)

What does that mean?

Berkeley 5th December 2016
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

\[
\text{com} ::= X \to \text{exp} \mid \text{com}_1;\text{com}_2 \mid \text{if} \ bexp \ \text{then} \ \text{com}_1 \ \text{else} \ \text{com}_2
\]

\mid \text{while} \ bexp \ \text{do} \ \text{com} \ \text{od}

- A *state* is a map from variables to values.
- A command is *interpreted* as a *partial* function from states to states.
- \([\text{com}] : \text{State} \to \text{State}\)
- \([\cdot]\) 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

- 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.