Semantics of Nondeterminism: Functions, Strategies and Bisimulation

Paul Blain Levy, University of Birmingham

Outline of Talk

1. What is denotational semantics?
2. Nondeterminism and its challenges
3. Changing perspectives

Black boxes and stickers

Analogy Chips $\longrightarrow$ Engines $\longrightarrow$ Aeroplanes

Each item is enclosed in a black box, with a sticker on the outside.

The sticker tells us what that item does, but not how.

If we read the stickers on each component, we must be able to work out what to write on the sticker for the whole system. This is called compositionality.

It is safe to change the implementation of a component, provided the sticker information is left unchanged.

Putting meanings together

A denotational semantics is a sticker policy for a programming language.

The sticker on each program is called its denotation.

The semantics must specify

1. what information to include—too little is catastrophic, too much is bad
2. how to put denotations together.

Styles of denotational semantics

What kind of mathematical structure do we write on a sticker?

In functional semantics, a denotation is a function, describing what the program outputs for each possible input.

In game semantics, a denotation is a strategy, describing the interaction between a program and its environment.

Fruits of denotational research

Research in denotational semantics has fed into

programming languages Java generics, the ML family

programming idioms monads in functional programs

program logics parametricity, separation logic

compiler technology continuation-based compilation

verification tools model-checking, both from functional and game semantics

Nondeterministic languages

In modern languages, programs can be nondeterministic:

- a program has a set of possible behaviours
- but the programmer does not know its precise behaviour.

Example: programs using concurrent threads—the precise behaviour will depend on the scheduler.

Choices can be made at different stages at runtime:

- Erratic nondeterminism: choice before execution.
- Ambiguous nondeterminism: choice during execution.

Adapting Denotational Semantics

Researchers have attempted to apply the mathematical structures developed for deterministic languages to nondeterministic ones.

This has had some success. . .

- but does not work for all forms of nondeterminism (e.g. ambiguous)
- and, where it does work, leaves out some important information.

Safety and Liveness

Requirements for a customer service program:

Safety and Liveness

Requirements for a customer service program:

- The program must not kill the customer. Safety

Safety and Liveness

Requirements for a customer service program:

- The program must not kill the customer. Safety
- The program must greet the customer. Liveness

Safety and Liveness

Requirements for a customer service program:

- The program must not kill the customer. Safety
- The program must greet the customer. Liveness
- If the program insults the customer, it must apologize. Conditional liveness

Safety and Liveness

Requirements for a customer service program:

- The program must not kill the customer. Safety
- The program must greet the customer. Liveness
- If the program insults the customer, it must apologize. Conditional liveness
- The program must stop insulting the customer. Infinite liveness

Safety and Liveness

Requirements for a customer service program:

- The program must not kill the customer. Safety
- The program must greet the customer. Liveness
- If the program insults the customer, it must apologize. Conditional liveness
- The program must stop insulting the customer. Infinite liveness

The classical denotational semantics cannot recognize the last two properties.

Changing Perspectives

The longstanding problems of nondeterminism are starting to succumb.

- Roscoe introduced a functional semantics of conditional liveness (2005).
- Levy introduced a game semantics for infinite trace equivalence (2005).
- Levy showed that ambiguous nondeterminism does not have a functional semantics (2007).

Bisimulation

Recall A denotational semantics must specify

1. what information to include in a denotation
2. how to put denotations together.

Bisimulation is a way of answering the first question, by specifying when choices are made during execution.

Theorems of Howe (1998), Lassen (1999), Levy (2007) show this to be a good answer to the first question.

We need to answer the second question!

Research Themes

Fundamental research in the semantics of nondeterminism, covering the following areas.

- Affine type theory: finding the right calculus
- Algebraic structure
- Game semantics with bisimulation
- Analysis of computational adequacy: showing that the information provided by a denotation is accurate
- Functional semantics of bismulation
- Must testing and ambiguous nondeterminism
- Call-by-need and dataflow: relating research on nondeterminism to the theory of concurrency