Semantics of programming languages: an introduction

Matthew Hennessy

December 3, 2014

Formal methods

The application of maths and logic to Computer Science

- Checking that a program does what it is supposed to do
  - Tradition: use extensive test suites
  - The future: augment test suites with formal logic
- Mathematical models of system behaviour
  - Outcome: predict future behaviour using model checking
- Design of programming languages
  - describe exact behaviour of all programs from the language

Industrial interest

- Intel IBM Microsoft Google Oracle NASA ...
Course Aims:
To introduce students to formal methods for specifying the semantics of programming languages and formal techniques for verifying the behaviour of programs.

Learning Outcomes:
On completion of the course students will be able to:
- write operational semantics for simple sequential languages
- reason formally about the behaviour of simple sequential programs
- understand the basic principles underlying behavioural equivalences of systems

Programming language definition

Constituents:
- **Syntax**: what sequences of symbols are valid programs? Grammars, parsers, lexical analysers, . . .
- **Pragmatics**: Descriptions and examples of how features of the language should be used. Design issues. Implementation issues . . .
- **Semantics**: The meaning of language’s features; e.g. what should happen when a program is executed. When are two program fragments equivalent? When does a program satisfy a specification?

**Formal semantics**: The use of logic to describe what programs should do when executed.
Benefits of formal, mathematical semantics

- **Implementation**: machine-independent specification of behaviour. Correctness of compilers, including optimisations, static analyses . . . .
- **Verification**: semantics supports reasoning about programs, specifications and other properties, both mechanically and by hand.
- **Language design**: subtle ambiguities in existing languages often come to light, and cleaner, clearer ways of organising things can be discovered.

Code optimisation

- **Q**: When can code fragment \( C \) be replaced by code fragment \( D \) in a program \( P \) ?
- **Ans**: When they have the same semantics

Do these two Java methods have the same semantics?

```java
int add1(int x, int y)
{  return (x + y);
}

int add2(int x, int y)
{  return (y + x);
}
```
Informal descriptions

Here is the informal definition of

\[
\text{while } B \text{ do } C
\]


The command $C$ is executed repeatedly so long as the value of the expression $B$ remains true. The test takes place before each execution of the command.

Informal descriptions

An extract from the Algol 60 report:

*Finally the procedure body, modified as above, is inserted in place of the procedure statement and executed. If a procedure is called from a place outside the scope of any non-local quantity of the procedure body, the conflicts between the identifiers inserted through this process of body replacement and the identifiers whose declarations are valid at the place of the procedure statement or function designator will be avoided through suitable systematic changes of the latter identifiers.*

What is it saying?
Formal descriptions

Formal statements:

\[ \langle C, s_i \rangle \Downarrow s_t \]

Meaning:

*If program C is started in state \( s_i \) it will finish in state \( s_t \)*

Logic:

- What logical rules do these statements satisfy?
- How do these logical rules determine run-time behaviour of programs?

---

Formal descriptions

Formal statements:

\[ \langle C, s_i \rangle \Downarrow s_t \]

Meaning:

*If command C is started in state \( s_i \) it will finish in state \( s_t \)*

Logical rules: explained and used in course

\[
\begin{align*}
(B\text{-while}.T) & \quad \langle B, s \rangle \Downarrow \text{true} \\
(C, s) \Downarrow s_1 & \quad \langle \text{while } B \text{ do } C, s \rangle \Downarrow s_1 \\
\langle B, s \rangle \Downarrow \text{false} & \quad \langle \text{while } B \text{ do } C, s \rangle \Downarrow s' \\
\langle \text{while } B \text{ do } C, s \rangle \Downarrow s & \quad \langle \text{while } B \text{ do } C, s \rangle \Downarrow s'
\end{align*}
\]
Designing a new language

Syntax: See: http://esolangs.org/wiki/Brainfuck

Instructions in language \( B \) given by

\[
  I ::= \square | > | < | + | - | I_1 I_2 | [I]
\]

Semantics:

\( B \) programs manipulate arrays. Instructions \( > \) and \( < \) increment and decrement the index. Instructions \( + \) and \( - \) increment and decrement the value in the array at the current index. \( I_1 I_2 \) performs instructions \( I_1 \) and then instructions \( I_2 \). \([I]\) is a looping construct. Instruction \( \square \) does nothing.

Can we be more precise than this?

This course: operational semantics

We will consider:

- various styles of operational semantics
- a range of (simple) imperative and functional languages
- methods for proving facts about semantic definitions
- connections between various semantics

Maths:

- This course requires maths
- mathematical induction
- structural induction
Assessment

▶ Minor Assignments:
  ▶ Week 4
  ▶ Week 6
  ▶ Week 8

Major assessment:
  ▶ Week 11

Marking scheme:

▶ Assignment 1: 10%
▶ Assignment 2: 10%
▶ Assignment 3: 10%
▶ Assignment 4: 20%
▶ Exam: 50%