DD2457 Program Semantics and Analysis

Introductory Lecture

Lecture Outline

1. The lecturer
2. Introduction to semantics
3. Course syllabus
4. Course objectives
5. Course organization

Lecturer

- Dilian Gurov
- E-mail: dilian@csc.kth.se
- Phone: 08-790 81 98 (office)
- Visiting address:
  Osquars backe 2, floor 5, room 4520

- Research interests:
  – Program correctness:
    - Program models, logics, compositionality

What is ”*semantics*”?

- From the Greek: ”meaning”
- Linguistics: syntax, semantics, pragmatics
- Computer Science:
  – Programming languages (Java, Scheme)
  – Database query languages (SQL)
- ”Formal” semantics:
  meaning = behaviour
  as a **mathematical object!**

Program Behaviour: Aspects

- **Function**: what is the computed object?
- **Time**: how many computation steps?
- **Space**: how much memory is needed?
- **Interaction behaviour**:
  what sequences of **interactions**?
  on-going behaviour, reactive systems
  example: request server

Why Formal Semantics?

Provides the basis for formal analysis!

1. Programming language **implementation**
   - Abstract machines: unambiguous, correct
2. Program **transformation**
   - Program analysis: optimizing compilers
3. Program **correctness**
   - Program analysis: security
   - Program verification: proving correctness

The Example Language **While**

Euclid's algorithm for computing GCD

```
while ¬(x=y) do
   if x <= y
      then y := y - x
      else x := x - y
```

The Example Language **While**

- Integers and integer variables
- Arithmetic and boolean expressions
- Assignment, conditional, while loops
- Extensions:
  - Procedures
  - Exceptions
  - Concurrency

Semantic Styles

- **Operational** semantics:
  program behaviour: states & transitions
  granularity: big-step, small-step, …
- **Denotational** semantics:
  programs as "state transformers"
- **Axiomatic** semantics:
  programs as "predicate transformers"
  properties of states as logic assertions

Operational Semantics

- **Configurations**: "state of computation"
  intial and final configurations
- **Transitions** between configurations
  relation, defined by **rules**
- **Computations**:
  sequences of transitions
  from initial to final configurations

Operational Semantics

1. "Big-step" Operational Semantics
   transitions relate initial and final configs
2. "Small-step" Operational Semantics
   transitions relate internal configs
3. Abstract Machines
   execution strategy embedded into rules

Denotational Semantics

- Relates initial and final states
- Meaning of program is a mapping!
  from intial to final configurations,
  "state transformer"
- Meaning of constructs:
  given by defining equations
- Tricky for loops: recursive equations
  Fixed-point Theory, Domain Theory

Axiomatic Semantics

- Relates properties of initial and final states
  programs are ”predicate transformers”,
  properties of states: state assertions
- Meaning of programs: Hoare triples
- Meaning of constructs:
  given by rules (”axiomatic”)

Course Syllabus

Part I. Operational semantics and language implementation

Part II. Denotational semantics and program analysis

Part III. Axiomatic semantics and program verification

Course Aim

- Obtain an abstract understanding of program behaviour:
  - Leads to **better software design**
  - Enables to argue about **correctness**
- Apply and develop abstract tools
  - The basis for developing concrete tools
  - Two lab assignments

Intended Learning Outcomes

After the course, students should be able to:

- Motivate and relate the various semantic styles
- Extend these to other language features
- Apply the various techniques:
  - Prove correctness of implementation:
    spec: op. sem., impl: abstract machine, relate!
  - Perform various program analyses:
    detection of signs, error detection, security
  - Prove correctness of programs:
    construct proof tableaux, extract verification con’s

Course Organization

- 15 lectures/tutorials, 2 labs,
  6 assignments, 1 written exam
- Course **book**:
  ”*Semantics with Applications*” by Nielson and Nielson, check Kårbokhandeln
- Course **web page**:
  www.kth.se/social/course/DD2457/
- Course board: sv. ”kursnämnd”
  volunteers?

Assignments

- **6 written homework assignments**. Consist of exercises taken from the textbook or composed by the course leader
- Written solutions are checked in class by **peer reviewing**, with the help of solutions provided by the course leader; are then collected
- Each peer reviewed and passed assignment gives a **bonus point** for the final exam (out of 30 points). Criterion for passing is to have made a decent attempt to solve the problem rather than to have necessarily produced a correct solution

Lab Assignments

- **2 laboratory assignments**. Carried out in teams of at most two. The lab sessions are only for getting assistance and presenting the solutions.
  1. Implementing an interpreter for the **While** language based on the notion of *abstract machines* developed in Chapter 4 of the textbook.
  2. Adapting the interpreter from the first one to run with abstract values instead of with concrete ones, thus implementing the *abstract interpretation* technique for program analysis developed in Chapter 7 of the textbook, but in an operational semantics style. The technique is applied to *program transformation*.