CS520 Theory of Programming Languages

Introduction

Hongseok Yang
KAIST

How to analyze programming languages (their constructs, type systems, implementations, etc) formally?

We will study mathematical tools for doing such analysis.

Preview 1:
Abstract syntax

- What is a program? What kind of syntactic object is it?

Preview 1:
Abstract syntax

- What is a program? What kind of syntactic object is it?
- Bad answer: a sequence of characters.

Preview 1:
Abstract syntax

- What is a program? What kind of syntactic object is it?
- Bad answer: a sequence of characters.
- Our answer: an instance of an abstract syntax.
- Mathematically, an element of an initial algebra.

Preview 2: Domain theory

```
>>> def F(g): return g
...
```

- Which mathematical object does the program F denote?

Preview 2:
Domain theory

```
>>> def F(g): return g
...
```

- Which mathematical object does the program F denote?
- Identity function in [D→D] for some D.

Preview 2: Domain theory

```
>>> def F(g): return g
...
>>> F(F)
<function F at 0x10c573410>
```

- Which mathematical object does the program F denote?
- Identity function in [D→D] for some D.
- But D should include [D→D]. Impossible if D is a set.

Preview 2:
Domain theory

```
>>> def F(g): return g
...
>>> F(F)
<function F at 0x10c573410>
```

- Which mathematical object does the program F denote?
- Identity function in [D→D] for some D.
- But D should include [D→D]. Impossible if D is a set.
- Possible if D is a domain & [D→D] has only continuous fns.

Preview 3: Evaluation order

```
>>> def f(x): return (x+x)
...
>>> f(f(3))
12
```

- Should we compute f(3) before applying f to f(3)?

Preview 3: Evaluation order

```
>>> def f(x): return (x+x)
...
>>> f(f(3))
12
```

- Should we compute f(3) before applying f to f(3)?
- Yes. Eager evaluation. Python, OCaml, Scheme, etc.
- No. Normal-order evaluation or lazy evaluation. Haskell.

Preview 3:
Evaluation order

```
>>> def f(x): return 3
...
>>> f(f(3))
12
```

- Should we compute f(3) before applying f to f(3)?
- Yes. Eager evaluation. Python, OCaml, Scheme, etc.
- No. Normal-order evaluation or lazy evaluation. Haskell.

Preview 3:
Evaluation order

```
>>> def f(x): return 3
...
>>> f(f(3))
12
```

- Should we compute f(3) before applying f to f(3)?
- Yes. Eager evaluation. Python, OCaml, Scheme, etc.
- No. Normal-order evaluation or lazy evaluation. Haskell.
- To be analysed via operational and denotational semantics.

Preview 4:
Type system

```
import typing
from typing import Callable

def twice(f : Callable[[int],int], x : int) -> int:
  return(f(f(x)))
```

- Types help develop correct programs.

Preview 4:
Type system

```
import typing
from typing import Callable

def twice(f : Callable[[int],int], x : int) -> int:
  return(f(f(x)))
```

- Types help develop correct programs.
- Can we infer types automatically?
- What mathematical objects do types denote?

Preview 4:
Type system

```
import typing
from typing import Callable

def twice(f : Callable[[int],int], x : int) -> int:
  return(f(f(x)))
```

- Types help develop correct programs.
- Can we infer types automatically? Type inference algo.
- What mathematical objects do types denote? Partial equivalence relation.

- Predicate Logic (Ch1).
- The Simple Imperative Language (Ch2).
- Program Specification and Their Proofs (Ch3).
- Failure, Input-Output, and Continuation (Ch5).
- Transition Semantics (Ch6).
- An Introduction to Category Theory (Tennent Ch8).
- Recursively-Defined Domains (Tennent Ch10).
- The Lambda Calculus (Ch10).
- An Eager Functional Language (Ch11).
- Continuation in a Functional Language (Ch12).
- A Normal-Order Language (Ch14).
- The Simple Type System (Ch15).

Imperative Languages

- Predicate Logic (Ch1).
- The Simple Imperative Language (Ch2).
- Program Specification and Their Proofs (Ch3).
- Failure, Input-Output, and Continuation (Ch5).
- Transition Semantics (Ch6).
- An Introduction to Category Theory (Tennent Ch8).
- Recursively-Defined Domains (Tennent Ch10).
- The Lambda Calculus (Ch10).
- An Eager Functional Language (Ch11).
- Continuation in a Functional Language (Ch12).
- A Normal-Order Language (Ch14).
- The Simple Type System (Ch15).

Imperative Languages

- Predicate Logic (Ch1).
- The Simple Imperative Language (Ch2).
- Program Specification and Their Proofs (Ch3).
- Failure, Input-Output, and Continuation (Ch5).
- Transition Semantics (Ch6).
- An Introduction to Category Theory (Tennent Ch8).
- Recursively-Defined Domains (Tennent Ch10).
- The Lambda Calculus (Ch10).
- An Eager Functional Language (Ch11).
- Continuation in a Functional Language (Ch12).
- A Normal-Order Language (Ch14).
- The Simple Type System (Ch15).

Functional Languages

Imperative Languages

- Predicate Logic (Ch1).
- The Simple Imperative Language (Ch2).
- Program Specification and Their Proofs (Ch3).
- Failure, Input-Output, and Continuation (Ch5).
- Transition Semantics (Ch6).

Math tools

- An Introduction to Category Theory (Tennent Ch8).
- Recursively-Defined Domains (Tennent Ch10).
- The Lambda Calculus (Ch10).
- An Eager Functional Language (Ch11).
- Continuation in a Functional Language (Ch12).
- A Normal-Order Language (Ch14).
- The Simple Type System (Ch15).

Functional Languages

Course webpage

https://github.com/hongseok-yang/graduatePL20

- Primary source of information about the course.

Blackboard lectures

- Nearly all the lectures will use blackboard, not slides.
- My handwritten notes will be available in the course webpage.

Evaluation

- Final exam — 40%.
- Homework (4 problem sheets) — 30%.
- Two critical surveys — 30%.

Evaluation

- Final exam — 40%.
- Homework (4 problem sheets) — 30%.
- Two critical surveys — 30%.

Final exam

- 10 hour take-home exam.
- From 10am on 12 Dec (Sat) until 8pm on 12 Dec (Sat).
- The exam paper will be distributed in KLMS.
- Solutions should be submitted in KLMS.

Evaluation

- Final exam — 40%.
- Homework (4 problem sheets) — 30%.
- Two critical surveys — 30%.

Critical surveys

- Study an assigned topic for yourself.
- Write a review (up to 3 pages) excluding bibliography.
- Try to go beyond simple survey. Your own thoughts. Connection with other PL concepts. In-depth study.
- Writing (20%). Understanding (40%). Originality (40%).

Critical survey 1

- Deadline: 30 Oct (Friday). By 23:59.
- Topic: Concurrent separation logic.
- Look at the course webpage for guideline.

Critical survey 2

- Deadline: 7 Dec (Monday). By 23:59.
- Topic: Relational parametricity.
- Look at the course webpage for guideline.

Honour code

- We adopt a strict policy for handling plagiarism and academically dishonest behaviours.
- A student will get F if
  - she or he is found to copy texts from papers and books without rephrasing them properly; or
  - he or she is found to cheat in an exam or copy answers or code from friends' or other sources.

Teaching staffs

- Prof Hongseok Yang (Lecturer). hongseok00@gmail.com. Office hour: 6pm-7pm at ZOOM.
- Mr Hyoungjin Lim (TA1). lmkmkr@kaist.ac.kr
- Mr Dongwoo Oh (TA2). dongwoo@kaist.ac.kr
- TAs' office hours will be announced shortly.