

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 $[D \rightarrow D]$ 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 $[D \rightarrow D]$ for some D .
- But D should include $[D \rightarrow D]$. 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 $[D \rightarrow D]$ for some D .
- But D should include $[D \rightarrow D]$. Impossible if D is a set.
- Possible if D is a **domain** & $[D \rightarrow D]$ 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 4
...
>>> f(f(3))
4
```

- 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 4
...
>>> f(f(3))
4
```

- 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**.

Related topic 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.

Related topic 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?

Related topic 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).

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).

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).

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).

- An Introduction to Category Theory (Tennent Ch8). **Math tools**
- Recursively-Defined Domains (Tennent Ch10).

- The Lambda Calculus (Ch10).
- An Eager Functional Language (Ch11).
- Continuation in a Functional Language (Ch12).

Functional Languages

Course webpage

<https://github.com/hongseok-yang/graduatePL21>

- Primary source of information about the course.

Baby Edu4.0

- Pre-recorded video lectures from the last year.
- Which video lectures to watch in each week? How to access them? See the course webpage.
- Q&A sessions on Wednesdays. Write questions at KLMS as replies to discussion topics in a KLMS forum.
- Four oral tests on 03/29, 04/19, 05/17, 06/17 (Mondays).

Evaluation

- Homework (4 problem sheets) — 40%.
- ZOOM oral tests — 30%.
- Two critical surveys — 30%.

Evaluation

- Homework (4 problem sheets) — 40%.
- ZOOM oral tests — 30%.
- Two critical surveys — 30%.

Four ZOOM oral tests

- Randomised in-class oral tests.
- 13:00-14:15 (03/29,05/17) and 13:00-14:30 (04/19,06/14).
- Attendance check at 13:30.
 - If a student is asked a question before the check but is absent, the student is regarded as missing the test.
- Expected to answer over microphone. Also, may ask to turn on video camera. See the course webpage for detail.

Evaluation

- Homework (4 problem sheets) — 40%.
- ZOOM oral tests — 30%.
- Two critical surveys — 30%.

Critical survey

- 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: 28 April (Wednesday). By 23:59.
- Topic: Concurrent separation logic.
- Look at the course webpage for guideline.

Critical survey 2

- Deadline: 7 June (Monday). By 23:59.
- Select a paper in POPL'19-21 and PLDI'19-20, and write a critical survey about the topic of the paper.
- Look at the course webpage for guideline.

Honour code

- We adopt a strict policy for handling plagiarism and dishonest academic behaviours.
- A student will get F if
 - she or he is found to copy texts from papers and books without paraphrasing them properly; or
 - he or she is found to cheat in a test 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 Sangho Lim (TA1). lim.sang@kaist.ac.kr
- Mr Dongwoo Oh (TA2). dongwoo@kaist.ac.kr
- TAs' office hours will be announced shortly.