I Introduction

Due to his grammar mistake, Wilbur found a position. It just wasn’t the one he wanted.

Will work as food
I Introduction

The role of PLs in CS

How to study PLs?

Topics & Schedule
Science and Engineering

Science

Understand & Explain

Builds on

Engineering

Builds on

enables

Physics
Chemistry
Theoretical CS

ME, EE, ...
CE, ...
SE, ...

Introduction
What is Computation?

**Systematic Transformation of Representation**

- **Systematic**
  - Intensional
  - Description

- **Transformation**
  - Function

- **Representation**
  - Abstraction that preserves particular features

- **Description** given in a **Language**
Facilitating Computation by Machines

Programming Languages

Programming Language: Formal Description of Computation

Many users/domains + many machines ⇒ Many PLs
Impact of Programming & Language Design

- Design a Language
- Develop a Program
- Use a Program

Multiplier
Multiplier

Empower people to ...
Empower people to ...
Faster and more reliable problem solving
I Introduction

The role of PLs in CS

How to study PLs?

Topics & Schedule
Language Diversity

- Perl
- Ruby
- Python
- Haskell
- Java
- Scheme
- Racket
- Fortran

Introduction
Exploring Languages

Comparing by programming?

Describing common concepts

Introduction
PL Concept Hierarchy

Set

Elements: strings over alphabet

Functional Language

Domain: Structured Language

Semantics

Domain & Range: Structured Language

Structured Language

Domain: Structured Language

Representation

Range: Structured Language

Computation

Range of Semantics:

Programming Language

CS 581

CS 582

CS 585

Domain-Specific Languages

Formal Language

CS 321

CS 521

Elements: typed trees

Function

CS 582
Languages Landscape

Metalanguages

Syntax (form)

Semantics (meaning)

Properties

Paradigm (feature sets)

Language Aspects

Values

Operations

Names

Functions

Data types

State

Typing

Objects

Grammar

Rule systems

Haskell

Idris

Languages Landscape

Introduction

Grammars

Rule systems

...
Programming Language Descriptions

Math
- (Abstract) Syntax
- Grammar
- Function
- Rule System
- Theorem & Proof

Programming Language
- Semantics
- Denotational
- Operational
- Type System
- Properties

Haskell
- Data Type
- Function
- –

Idris
- Data Type
- Function
- Dependent Type
- Dependent Type
- Dep.Type & Tree
I Introduction

The role of PLs in CS

How to study PLs?

Topics & Schedule
Teaching Approach

Language Concepts
- values
- operations
- state
- names
- functions
- data types
- typing
- objects

Example Language

Language Aspects
- syntax
- semantics
- typing
- properties

Metalanguages
- Math
- Haskell
- Idris

Illustrate a concept by defining its aspects as part of an example language using a metalanguage.
Example Languages

Language Concepts:
- values
- operations
- state
- names
- functions
- data types
- typing
- objects

Expression Languages:
- Num, Bool, NumBool

Stateful Languages:
- Stack, Asm, Imp

Languages with Bindings:
- Lambda Calculus

Type Systems:
Language Definitions

Concept

Definition Profile

Language

Aspect

Concept

Metalanguage

Aspect

Metalanguage
# Some Example Definitions

<table>
<thead>
<tr>
<th>Conditional</th>
<th>NumBool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Grammar</td>
</tr>
</tbody>
</table>

$t \in \text{Term} ::= \ldots \mid \text{if } t \text{ then } t \text{ else } t$

data Term = \ldots
\mid \text{If Term Term Term Term}

data Term : Type where \ldots

<table>
<thead>
<tr>
<th>Conditional</th>
<th>NumBool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Haskell</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Conditional</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Grammar</td>
</tr>
</tbody>
</table>

$e \in \text{Expr} ::= \ldots$

$\mid \text{if } \{e\} \{s\} \text{ [else } \{s\}]$

data Expr = \ldots

data Stmt = \ldots
\mid \text{If Expr Stmt (Maybe Stmt)}
More Definitions

Data Type:

```haskell
data Value = B Bool | I Int
```

Semantic Function:

```haskell
sem :: Term \rightarrow Value
sem (If c t e) | sem c == B True = sem t
               | otherwise = sem e
```

Big-Step Operational Semantics:

```
t_1 \Downarrow \text{true} \quad t_2 \Downarrow t \\
\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \Downarrow t
```

```
t_1 \Downarrow \text{false} \quad t_3 \Downarrow t \\
\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \Downarrow t
```

Conditional Typing Rules:

```
t_1 : \text{Bool} \quad t_2 : T \quad t_3 : T \\
\text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T
```
More Definitions

\[ \text{data (\(\rightarrow\)) : Term \rightarrow Term \rightarrow Type} \]

where

\[ \text{IfT} : c \rightarrow T r u \rightarrow t \rightarrow v \rightarrow \]

\[ (\text{If } c \text{ Then } t \text{ Else } e) \rightarrow v \]

\[ \text{IfF} : c \rightarrow F l s \rightarrow e \rightarrow v \rightarrow \]

\[ (\text{If } c \text{ Then } t \text{ Else } e) \rightarrow v \]

\[ \begin{array}{c}
\text{Conditional} \\
\text{Semantics} \\
\text{Idris}
\end{array}\]

\[ \begin{array}{c}
\text{NumBool} \\
\text{Rules}
\end{array}\]

\[ t_1 \downarrow \text{true} \quad t_2 \downarrow t \]

\[ \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \downarrow t \]

\[ t_1 \downarrow \text{false} \quad t_3 \downarrow t \]

\[ \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \downarrow t \]
Crucial Representations

Syntax → Data Type → Haskell & Idris

Rules & Semantics → Dependent Type → Idris

Theorems →
Schedule & Grading

Tentative Lecture Syllabus

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
<th>Thu Quiz</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction, Idris</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>Dependent Types, Proofs</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>Idris Practice</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>Inference Rules</td>
<td>✓</td>
</tr>
<tr>
<td>5</td>
<td>Operational Semantics</td>
<td>–</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Midterm</td>
</tr>
<tr>
<td>7</td>
<td>Operational Semantics</td>
<td>✓</td>
</tr>
<tr>
<td>8</td>
<td>Language Properties</td>
<td>✓</td>
</tr>
<tr>
<td>9</td>
<td>Type Systems &amp; Polymorphism</td>
<td>–</td>
</tr>
<tr>
<td>10</td>
<td>Type Inference</td>
<td>–</td>
</tr>
</tbody>
</table>

Important Dates and Times

<table>
<thead>
<tr>
<th>Quizzes</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jan 18</td>
<td>Thu, 10:00am – 10:20am</td>
</tr>
<tr>
<td></td>
<td>Feb 1</td>
<td></td>
</tr>
<tr>
<td>Midterm Exam</td>
<td>Feb 15</td>
<td>Thu, 10:00am – 11:00am</td>
</tr>
<tr>
<td>Quizzes</td>
<td>Feb 22</td>
<td>Thu, 10:00am – 10:20am</td>
</tr>
<tr>
<td></td>
<td>Mar 1</td>
<td></td>
</tr>
<tr>
<td>Final Exam</td>
<td>Mar 22</td>
<td>Wed, 2pm – 3:50pm</td>
</tr>
</tbody>
</table>

Grading

- 10% Class participation (during class and on Piazza)
- 20% Quizzes
- 30% Midterm Exam
- 40% Final Exam