I Introduction

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

The role of PLs in CS

How to study PLs?

Topics & Schedule
Science and Engineering

Understand & Explain

builds on

enables

Science

Build

Engineering

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
Multiplication

\[ \log(x \cdot y) = \log x + \log y \]
Origami

How to make an origami crane

1. Fold the paper in half diagonally.
2. Fold the paper in half widthwise.
3. Unfold the paper and make creases on both sides.
4. Fold the corners to the center.
5. Make a mountain crease on the diagonal.
6. Repeat steps 4 and 5 on the other side.
7. Fold the paper in half along the crease made in step 2.
8. Unfold the paper and make creases on both sides.
9. Fold the paper in half along the crease made in step 7.
10. Unfold the paper and make creases on both sides.
11. Fold the paper in half along the crease made in step 9.
12. Unfold the paper and make creases on both sides.
13. Fold the paper in half along the crease made in step 11.
14. Unfold the paper and make creases on both sides.
15. Fold the paper in half along the crease made in step 13.
16. Unfold the paper and make creases on both sides.
17. Fold the paper in half along the crease made in step 15.
18. Unfold the paper and make creases on both sides.
19. Fold the paper in half along the crease made in step 17.

Folding
Transformation
Paper
Objects
Representation

Introduction
Summary: PL ↔ CS

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

Multiplier

Design a Language → Empower people to ...

Multiplier

Develop a Program → Empower people to ...

Multiplier

Use a Program → 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
Fortran
Haskell
Java
Scheme
Racket
Smalltalk
Exploring Languages

Comparing by programming?

Describing common concepts
PL Concept Hierarchy

Formal Language
- CS 321
- CS 521

Structured Language
- CS 585
  Domain-Specific Languages

Semantics
- Computation
- Representation
- Domain & Range
- Domain: Structured Language
- Range: Structured Language

Set
- Elements: strings over alphabet

Function
- Elements: typed trees

Programming Language
- CS 581
- CS 582

Introduction
Languages Landscape

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

Grammars
- Rule systems
- ...

Haskell
- Idris
- ...

Metalanguages

Language Aspects
- syntax (form)
- semantics (meaning)
- properties
- paradigm (feature sets)

Language Processing
- parsing
- type checking
- compiling
- interpreting

(“Engineering”, CS 480)
Programming Language Descriptions

Math
- Grammar
- Rule System
- Theorem & Proof

Haskell
- Data Type
  - Function

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

Programming Language
- (Abstract) Syntax
- Syntax
- Semantics
- Denotational
- Operational
- Type System
- Properties
I Introduction

The role of PLs in CS

How to study PLs?

Topics & Schedule
Teaching Approach

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

Expression Languages
- Num, Bool, NumBool

Stateful Languages
- Stack, Asm, Imp

Languages with Bindings
- Lambda Calculus

Type Systems

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

Concept

Definition

Language

Aspect

Metalanguage

Definition Profile

Concept | Language
Aspect | Metalanguage
Some Example Definitions

### Conditional Syntax

$t \in \text{Term} ::= \ldots$

$|$ $\text{if } t \text{ then } t \text{ else } t$

### Conditional Grammar

\[ \text{data } \text{Term} = \ldots \]

$|$ \text{If Term Term Term}$

### NumBool Syntax

\[ \text{data } \text{Term} : \text{Type where } \ldots \]

\[ \text{If} : \text{Term} \rightarrow \text{Term} \rightarrow \text{Term} \rightarrow \text{Term} \]

### Conditional Haskell

\[ \text{data } \text{Term} = \ldots \]

$|$ \text{If Term Term Term}$

### Haskell Syntax

\[ \text{data } \text{Term} : \text{Type where } \ldots \]

\[ \text{If} : \text{Term} \rightarrow \text{Term} \rightarrow \text{Term} \rightarrow \text{Term} \]

### Java Syntax

\[ e \in \text{Expr} ::= \ldots \]

\[ s \in \text{Stmt} ::= \ldots \]

$|$ \text{if } (e) \{s\} \text{ [else } \{s\}]$

### Idris Syntax

\[ \text{data } \text{Term} : \text{Type where } \ldots \]

### Java Grammar

\[ \text{data } \text{Expr} = \ldots \]

\[ \text{data } \text{Stmt} = \ldots \]

$|$ \text{If Expr Stmt} (\text{Maybe Stmt})
More Definitions

### Data Types

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

### Semantic Function

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

### Typing Rules

```haskell
\[
\frac{t_1 : B \text{ Bool}}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T}
\]
```

### Operational Semantics

```haskell
\[
\frac{t_1 \Downarrow \text{true} \quad t_2 \Downarrow t}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \Downarrow t}
\]
\[
\frac{t_1 \Downarrow \text{false} \quad t_3 \Downarrow t}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \Downarrow t}
\]
```

### Denotational Semantics

Conditional NumBool Semantics

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

Big-Step Operational Semantics
More Definitions

```
data (=>) : Term → Term → Type where
  ...
  IfT : c => Tru → t => v →
       --------------------------
       (If c Then t Else e) => v
  IfF : c => Fls → e => v →
       --------------------------
       (If c Then t Else e) => v
```

```
<table>
<thead>
<tr>
<th>Conditional</th>
<th>NumBool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantics</td>
<td>Idris</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Conditional</th>
<th>NumBool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantics</td>
<td>Rules</td>
</tr>
</tbody>
</table>

```

\[
\begin{align*}
& t_1 \Downarrow \text{true} \quad t_2 \Downarrow t \\
& \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \Downarrow t \\
& t_1 \Downarrow \text{false} \quad t_3 \Downarrow t \\
& \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \Downarrow t
\end{align*}
\]

```

Introduction
Crucial Representations

Syntax → Data Type

Rules & Semantics → Dependent Type

Theorems → Dependent Type

Haskell & Idris

Idris
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>Operational Semantics</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>Date</th>
<th>Assignment</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 19</td>
<td>Quizzes</td>
<td>Thu, 10:00am – 10:20am</td>
</tr>
<tr>
<td>Feb 2</td>
<td>Midterm Exam</td>
<td>Thu, 10:00am – 11:00am</td>
</tr>
<tr>
<td>Feb 23</td>
<td>Quizzes</td>
<td>Thu, 10:00am – 10:20am</td>
</tr>
<tr>
<td>Mar 2</td>
<td>Final Exam</td>
<td>Wed, 2pm – 3:50pm</td>
</tr>
</tbody>
</table>

Grading

<table>
<thead>
<tr>
<th>Grade</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>Class participation (during class and on Piazza)</td>
</tr>
<tr>
<td>20%</td>
<td>Quizzes</td>
</tr>
<tr>
<td>30%</td>
<td>Midterm Exam</td>
</tr>
<tr>
<td>40%</td>
<td>Final Exam</td>
</tr>
</tbody>
</table>

Note on Quizzes and Exams:

for Students with Disabilities.

in advance of the test. Students seeking accommodations should be registered with the Office of Services

of the term. In order to arrange alternative testing the student should make the request at least one week

information the instructor should know, or who need special arrangements in the event of evacuation,

Homework will not be graded. I may discuss selected exercises in class upon request.

for the midterm and final exams (the problems on the exams will be easier than the homework problems).

Homework is optional. Working through the homework problems is a good preparation

one-page cheat sheet

Schedule

Week | Topic                                      | Thu Quiz |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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>Operational Semantics</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>