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

Science

Build

Engineering

builds on
enables

Physics
Chemistry
Theoretical CS

Builds on

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

Introduction
What is Computer Science?

“Computer science is no more about computers than astronomy is about telescopes.”

Edsger Dijkstra (?

Computer Science = The Science of Computing
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 bottom left and top right corners of the paper to the center.
2. Fold the top and bottom edges to the center.
3. Fold the top and bottom edges to the center again.
4. Fold the top and bottom edges to the center one more time.
5. Open the paper and unfold the folds.
6. Fold the paper in half along the creases.
7. Fold the top and bottom edges to the center.
8. Fold the top and bottom edges to the center again.
9. Fold the top and bottom edges to the center one more time.
10. Open the paper and unfold the folds.
11. Fold the paper in half along the creases.
12. Fold the top and bottom edges to the center.
13. Fold the top and bottom edges to the center again.
14. Fold the top and bottom edges to the center one more time.
15. Open the paper and unfold the folds.
16. Fold the paper in half along the creases.
17. Fold the top and bottom edges to the center.
18. Fold the top and bottom edges to the center again.
19. Fold the top and bottom edges to the center one more time.
Summary: PL ↔ CS

Computing
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

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
Fortran
Scheme
Racket

Introduction
Exploring Languages

Comparing by programming?

Describing common concepts
Languages Landscape

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

Grammar
- Rule systems
- ...

Haskell
- Idris
- ...

Metalanguages

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

Language Processing
- parsing
- type checking
- compiling
- interpreting

Introduction

(“Engineering”, CS 480)
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

Language Aspects
- syntax
- semantics
- typing
- properties

Example Language

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:
- Languages with Bindings

Type Systems:
- Lambda Calculus
# Language Definitions

<table>
<thead>
<tr>
<th>Concept</th>
<th>Language</th>
<th>Aspect</th>
<th>Metalanguage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition Profile</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Introduction**

**Concept**

**Language**

**Aspect**

**Metalanguage**
Some Example Definitions

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

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

\[ e \in \text{Expr} ::= \ldots \]
\[ s \in \text{Stmt} ::= \ldots \mid \text{if } (e) \{s\} \mid \text{else } \{s\} \]

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

data Value = B Bool | I Int

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

Condition | NumBool | Haskell
---|---|---
if t_1 then t_2 else t_3 | T | T

Denotational Semantics

Big-Step Operational Semantics

<table>
<thead>
<tr>
<th>Condition</th>
<th>NumBool</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>if t_1 then t_2 else t_3</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>t_1 \Downarrow \text{true} &amp; t_2 \Downarrow t &amp; t \Downarrow t</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t_1 \Downarrow \text{false} &amp; t_3 \Downarrow t &amp; t \Downarrow t</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Introduction
More Definitions

data (\Rightarrow) : \text{Term} \to \text{Term} \to \text{Type} \text{ where }

\text{IfT} : 
  c \Rightarrow \text{Tru} \to \text{t} \Rightarrow \text{v} \to
  \begin{array}{l}
    \text{(If c Then t Else e)} \Rightarrow \text{v}
  \end{array}

\text{IfF} :
  c \Rightarrow \text{Tru} \to \text{e} \Rightarrow \text{v} \to
  \begin{array}{l}
    \text{(If c Then t Else e)} \Rightarrow \text{v}
  \end{array}

\begin{array}{l}
  \text{Condition} \ \text{NumBool} \\
  \text{Semantics} \ \text{Idris}
\end{array}

\begin{array}{l}
  \text{Condition} \ \text{NumBool} \\
  \text{Semantics} \ \text{Rules}
\end{array}

\begin{array}{l}
  t_1 \Downarrow \text{true} \quad t_2 \Downarrow t \\
  \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \Downarrow t
\end{array}

\begin{array}{l}
  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{array}
Crucial Representations

Syntax → Data Type

Rules & Semantics → Dependent Type

Theorems → Haskell & Idris

Idris
## Tentative Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction &amp; Idris</td>
</tr>
<tr>
<td>2</td>
<td>Dependent Types &amp; Proofs</td>
</tr>
<tr>
<td>3</td>
<td>Idris Practice</td>
</tr>
<tr>
<td>4</td>
<td>Inference Rules</td>
</tr>
<tr>
<td>5</td>
<td>Operational Semantics</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Language Properties</td>
</tr>
<tr>
<td>8</td>
<td>Lambda Calculus</td>
</tr>
<tr>
<td>9</td>
<td>Type Systems &amp; Polymorphism</td>
</tr>
<tr>
<td>10</td>
<td>Type Inference</td>
</tr>
</tbody>
</table>