CS 381: Programming Language Fundamentals
Course Introduction

January 4, 2016
Outline

Why study programming languages?
- Languages are at the heart of computer science
- Good languages really matter
- Language design can have a huge impact

How to study programming languages

Course logistics
What is computer science?

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

—Edsger Dijkstra

Computer Science = the science of computation
What is computation?

Computation is the **systematic transformation of representation**

- **Systematic**: according to a fixed plan
- **Transformation**: process that has a changing effect
- **Representation**: abstraction that encodes particular features

Languages play a central role:

- The “fixed plan” is an **algorithm**, which is described in a **language**
- Usually, the **representation** is also described in a **language**
What about software engineering?

<table>
<thead>
<tr>
<th>Science</th>
<th>Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>physics</td>
<td>structural engineering, …</td>
</tr>
<tr>
<td>chemistry</td>
<td>chemical engineering, …</td>
</tr>
<tr>
<td>“computing”</td>
<td>software engineering, …</td>
</tr>
</tbody>
</table>

Both are part of “computer science”
Central role of PL in CS

**Program**: a description of the plan to carry out and the representations it transforms

**Programming language**: a language for writing programs, i.e. describing computation

PL supports both aspects of CS:

- to understand and explain (science) we need languages to describe and reason about computations for ourselves
- to build cool stuff (engineering) we need languages to describe computations for a computer to execute
Outline

Why study programming languages?
- Languages are at the heart of computer science
- Good languages really matter
  - Language design can have a huge impact

How to study programming languages

Course logistics
Why good languages matter

The languages we use …
- influence our **perceptions**
- guide and support our **reasoning**
- enable and shape our **communication**

- What problems do we see? How do we reason about and discuss them?
- How do we develop, express, and share solutions?

*By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race.*

—Alfred North Whitehead via Kenneth Iverson’s ACM Turing Award Lecture, “Notation as a Tool of Thought”
Example: Positional number system

In the 13th century, this is how numbers were represented in Europe:

$$\text{MMCDXXXI} \div \text{XVII} = ? \:\text{:-(}$$

...even basic arithmetic is hard!

Fibonacci popularized the Hindu-Arabic notation

- didn’t just make arithmetic much more convenient …
- completely changed the way people thought about numbers, revolutionizing European mathematics
Example: Symbolic logic

For **over 2000 years** the European study of logic focused on syllogisms

*Every philosopher is mortal.*
*Aristotle is a philosopher.*
*Therefore, Aristotle is mortal.*

Only 256 possible forms … field solved!

A couple of **notational** innovations in the 19th century cracked it wide open

- George Boole – Boolean algebra
- Gottlob Frege – *Beggriffsschrift* (symbolic predicate logic)
Example: Feynman diagrams

Interactions of subatomic particles lead to brain-melting equations

- reasoning about interactions requires complex math
- high overhead to communicating problems and solutions

Only a handful of people can do this stuff!

In 1948, Richard Feynman introduced a visual language for representing interactions

- Raises level of abstraction
  - eliminates incidental complexity (math)
  - focus on essential complexity (interactions)
  - supports communication, collaboration (undergrads can do it)
Since the languages we use matter so much …

You better know how to *choose* the right one!

…or how to *create* it if it doesn’t exist!
Outline

Why study programming languages?
Languages are at the heart of computer science
Good languages really matter
Language design can have a huge impact

How to study programming languages

Course logistics
### Impact of programs and languages

<table>
<thead>
<tr>
<th>If you can ...</th>
<th>then you can ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>run a program</td>
<td>do something faster and more reliably</td>
</tr>
<tr>
<td>write a program</td>
<td>empower others to do new things</td>
</tr>
<tr>
<td>create a language</td>
<td>unlock potential for solving harder problems and bring the power of computing to new domains</td>
</tr>
</tbody>
</table>

Each level is a **multiplier** for potential impact!
Domain-specific languages

\[ F = ma \]

\[ E = mc^2 \]

2H₂ + O₂ → 2H₂O

Why study programming languages?
Outline

Why study programming languages?
Languages are at the heart of computer science
Good languages really matter
Language design can have a huge impact

How to study programming languages

Course logistics
One idea: just try out a bunch of languages

Not this course!
Analogy: choosing or designing a vehicle

There are many things you can’t learn by driving a bunch of them.
Our main focus: programming language concepts

Focus on how to define programming languages

For several toy languages, we will:

- define the **structure** of its programs
- define the **meaning** of its programs
- identify the **features** that are common to many languages
Role of metalanguages

**Metalanguage**: a language to define the structure and meaning of another language!

In this course:
- grammars
- Haskell
- English

Important metalanguage not covered:
- mathematics (CS 589 → CS 581)
- inference rules (CS 581 → CS 582)
On the other hand ... 

The only way to *really* learn to drive a bulldozer ... 

... is to *drive a bulldozer*!
Summary of our strategy

Focus mostly on programming language concepts

1. define abstract syntax of languages
2. define semantics of languages
   - scoping
   - parameter passing
   - exceptions
3. define type systems for languages

We use Haskell as a metalanguage for describing these concepts!

Introduce you to two new programming paradigms

1. functional programming (Haskell) – lots of practice
2. logic programming (Prolog) – last couple of weeks
Outline

Why study programming languages?
- Languages are at the heart of computer science
- Good languages really matter
- Language design can have a huge impact

How to study programming languages

Course logistics
You probably do not need to take this class!

There is a new version of **CS 581**:
- duplicates much of the material in CS 381
- open to all graduate students
- satisfies your undergrad PL requirement