Fundamentals of programming languages

Amer Diwan

Goals

- **To get a deep understanding of “modern” programming languages**
  - What are the important concepts underlying language features?
  - What are the strengths and weaknesses of the features?
  - What are the implications (performance and otherwise) of the features?
Prerequisites

- An introductory programming languages course
- Programming experience with an object-oriented programming language
- A willingness to spend a lot of energy reading papers and books and participating in discussions!

Reading material

- Papers from journals and conferences
What are the features that differentiate modern programming languages?

- Types
- Polymorphism
- Control constructs
- Memory management
- Module system
- ...

Types

- What are the primitive and aggregate types in languages?
- When is it legal to do an assignment? Pass a parameter?
- How strong is the type checking?
- How static is the type checking?
- When are two types equal?
- Objects/Classes... Need I say more?
Polymorphism

- What kinds of polymorphism does a language support?
- What are the performance implications of the polymorphism?

Control structures

- What local control constructs does the language provide?
- What non-local control constructs does the language provide?
- What are the implications of the constructs?
Memory management

- Does the language allow/encourage automatic memory management?
- What kinds of automatic memory management are required/allowed?
- What kinds of manual memory management are supported?

Aspects of programming language features

<table>
<thead>
<tr>
<th>Concepts</th>
<th>Concept implementation in language</th>
<th>Implementation in compiler and run-time system</th>
</tr>
</thead>
</table>

“Naive” view  “Ideal” view  “Realistic” view
Examples

<table>
<thead>
<tr>
<th>References and V-Tables</th>
<th>“Boxing”, and type inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclassing, method invocations in Java</td>
<td>Type variables in SML</td>
</tr>
<tr>
<td>Polymorphism</td>
<td></td>
</tr>
</tbody>
</table>

Why study concepts?

- Concepts are not burdened by language design or implementation considerations
- So it is easy to learn new languages
- So we are in a better position to understand the strengths and limitations of a language
Why study impact on compilers and run-time systems?

• It is part of the motivation behind many language designs
• So we understand the implication of using a language feature (or some combination of features)
• So we know what pitfalls to avoid when designing a language

Organization of the course

• Explore the following concepts:
  – types
  – polymorphism
  – control structures
  – memory management
• Look at exciting new trends in language design
Summary

• We can get a deeper understanding of language design and use by exploring three aspects of language features:
  – Important concepts
  – The different ways in which they are incorporated in languages
  – The implementation impact of the concepts

Next lecture

• Types
  – What are types?
  – Static and dynamic typing
  – Strong and weak typing
  – Type safety
• Readings
  – Mitchell handout, Sections 5.1, 5.2, 5.3