130 Wi 09

Last Lecture (!)

Final

- Next Tue 11:30 - 2:30 - Here
- practice problems and sample final up on the web
- quizzes and suggested hw are also practice problems
- Review session: Sunday Evening
  - Stay Tuned

End of Prolog

- Go to lecture-16.pl

Recap

Goat ~ Cabbage

Wolf

\( (F, G, W, C) \)

\( (\psiW, \psiG, \psiW, \psiC) \)
### Dimension: Syntax
- Languages have different syntax
  - But the difference in syntax can be superficial
  - C# and Java have different syntax, but are very similar
- We tried looking beyond superficial syntax, but...
  - indentation in Python

### Dimension: Computation model
- Functional: Lisp, OCaml, ML
- Imperative: Fortran, Pascal, C
- Object oriented: Smalltalk, C++, Java, C#
- Constraint-based: Prolog, CLP(R)

### Dimension: Memory model
- Explicit allocation-deallocation: C, C++
- Garbage collection: Smalltalk, Java, C#, OCaml, Python
- Does anybody here want explicit memory allocation? Why? Why not?
**Dimension: Typing model**

- Statically typed: Java, C, C++, C#, OCaml
  - saw some typing rules
  - saw how to infer types
  - formalizes the type system

- Dynamically typed: Lisp, Scheme, Perl, Smalltalk, Python
  - can make mistakes and only discover them much later

**Dimension: Execution model**

- Compiled: C, C++
- Interpreted: Perl, shell scripting PLs, Python
- Hybrid: Java

**What makes us like a language?**

- Type system?
- Convenient shortcuts (Perl)?
- Simplicity?
  - at odds with convenient shortcuts
- Orthogonality (language features don’t clash)
  - also at odds with convenient shortcuts
- Libraries? (very important one)
## Course material recap

1. Functional, **OCaml**, 4+ weeks
2. **OO**, **Python**, 3+ weeks
3. **Logic**, **Prolog**, 1+ weeks

### Functional

- No mutation! Instead have environments
- Higher order programming (functions as params and returned values)
- map and fold goodness
- type system and type inference

### Modular
Modularization

Nicely separated concerns:
- XML Parsing
- Pattern matching
- Collections
- Numerical processing

Language features allow for very nice separation

Limits of Modularization

Some concerns crosscut modular boundaries:
- Caching
- Logging
- Thread synchronization

The Goal of Aspect-Oriented Programming

- Write concerns in local aspects, and have them woven globally into the program by a weaver

Object Oriented
Python

- Dynamic dispatch
- Inheritance
- Decorators
- Streams, Metaclasses, ...

Constraint based programming

- What do you think of Prolog?
- Totally different model of computation
- Way cool, if you ask me!
- But can be hard to program in...

Prolog

- What do you think of Prolog?
- Totally different model of computation
- Way cool, if you ask me!
- But can be hard to program in...

Course Goals

“Free your mind”
-Morpheus

You will learn several new
- languages and constructs
- ways to describe and organize computation

Yes, you can do that in Java/Assembly but ...
- Use the right tool for the right job
Fun Stuff We Didn’t See ...

A new PL revolution due to three things:

- **Web/Concurrency:**
  - AJAX, JavaScript, Async Programming
  - “On the internet, no one knows you’re a dog”

- **Multicore/Concurrency:**
  - Cilk, Erlang, Haskell/STM (All functional)

- **Blogs (!):**
  - New communities grow around new PLs

If nothing else, remember this!

- Environment
- Closures
- Pattern-Matching
- Type Inference
- Namespaces
- Polymorphism
- ...