

APL: The Big Picture

Course Summary

- Programming Language Semantics
 - **Operational** – for defining program behavior
 - **Denotational** – for converting program to math
 - **Static** – for avoiding “stuck states” (bugs)
 - **Axiomatic** – for verifying program correctness
- Three styles of programming:
 - **Imperative** (programs are sequences of instructions)
 - **Functional** (programs are functions from inputs to outputs)
 - **Logic** (programs are declarative input-output relations)

Language Popularity

- Which languages are most popularly used in “real life” (i.e., industry)?
 - Unquestionably imperative ones (C/C++/Java)
- Why?
 - easy to compile (no longer a compelling reason)
 - momentum (well-developed tools, large labor pool)
 - easy to write code that almost works
- The “Software Crisis”
 - Microsoft spends >50% of its budget on testing (2008)
 - Their code still doesn’t work
 - “Find better programmers” is not the answer

Better Programming Languages

(What makes a language “advanced”?)

- Correctness over efficiency! (within reason)
 - “If I want it to run faster, I’ll buy more processors.”
 - Compilers as proof-assistants
- Elegant translation from mathematical spec to code
- Separation of concerns (the “what” vs. the “how”)
- Succinctness
 - Less code = fewer bugs
 - Code-reuse (parametric polymorphism)
- Modularity
 - Object-oriented programming
 - See also: OCaml module system, Aspect-oriented programming
- *Programmer efficiency* vs. program efficiency

Should we bury C/C++/Java?

- No! C/C++ is good for certain things:
 - writing the inner loop of a matrix multiplier
 - writing device drivers (but use formal verification)
 - implementing some runtime libraries (e.g., fast string libs)
- But can we please stop implementing entire software systems with it?
- Java was/is a great step forward...
 - brought type-safe programming to the masses
 - popularized automated garbage-collection
- But it still has major weaknesses
 - uncaught exceptions are only slightly better than crashes
 - language definition defies optimization

Grand Challenges

- What kind of language might segue the imperative programming world toward functional/declarative programming?
 - Example: F# [Syme et al., 2001]
- Can we use modern PL theory to debug/correct/analyze legacy codes?
 - Example: REINS [Wartell, Mohan, Hamlen, Lin; ACSAC 2012]
- Can we use PL theory to solve security problems like data confidentiality enforcement?
 - Example: Java Information Flow [Myers, POPL 1999]
- How can we create verified, highly parallelized software?
 - Example: *Verification of Parameterized Concurrent Programs By Modular Reasoning about Data and Control* [Farzan & Kincaid, POPL 2012]

Relevance/Usefulness

- Practical right now
 - Functional Programming
 - Operational/Denotational Semantics for compiler design and analysis
 - Type-checker design & implementation
- Not practical in itself, but fundamental for understanding real-world software verification
 - Lambda calculus
 - Hoare Logic
 - Structural Induction
- Learning how to write formal, rigorous proofs
 - essential if you want to do science, and not just programming
 - infrequently taught at the undergraduate level
 - if you can't prove easy things, you can't program hard things
 - every program is a constructive proof (Curry-Howard)
 - Example: if you can't reason inductively, you can't program recursively

Microsoft's Functional Language: F# (OCaml for Visual Studio .NET)

The screenshot shows a web browser displaying an article on eWEEK.com. The page layout includes a top navigation bar with the eWEEK.COM logo and links for 'SUBSCRIBE TO eWEEK', 'RSS Feeds', 'Print', and 'Newsletters'. Below this is a secondary navigation bar with categories like HOME, NEWS, REVIEWS, OPINIONS, STORAGE, SECURITY, CHANNEL, BLOGS, VIDEOS, PODCASTS, BUYERS GUIDE, CAREERS, and KNOWLEDGE. A breadcrumb trail reads: Home > Topics > Application Development > News > Microsoft Puts the 'F' in Functional.

The main article section features a large blue header for 'Application Development' and a sub-header 'Microsoft Puts the 'F' in Functional'. The author is identified as Darryl K. Taft, with a date of November 5, 2007. A 'TALKBACK' comment box is present, indicating it is the first comment on the article. A 'RELATED LINKS' sidebar on the right lists several related articles, including 'Microsoft Looking to Partners for SOA, 'Oslo'', 'Microsoft Models 'Oslo'', 'Microsoft Snags Open-Source Project Lead', 'Microsoft Pushes Cloud Development', and 'Microsoft Office Project Expands Its Reach'. A 'Share This' button and a 'Digg this' link are also visible.

Microsoft targets functional programming as a next big thing in software development.

MONTREAL—Microsoft is putting the "F" in functional programming with its F# language.

F#—pronounced "F sharp"—is a functional programming language out of Microsoft Research that the company will productize to target developers dealing with concurrency and those in the financial, scientific and technical and academic arenas.

ADVERTISEMENT

TAKE BACK CONTROL OF YOUR INFORMATION

Facebook's New Polymorphic, Statically-typed Web Dev Language

WIRED GEAR SCIENCE ENTERTAINMENT BUSINESS SECURITY DESIGN OPINION MAGAZINE

Facebook Introduces 'Hack,' the Programming Language of the Future

BY CADE METZ 03.20.14 | 12:00 PM | PERMALINK

[f Share](#) 98 [t Tweet](#) 49 [g+1](#) 594 [in Share](#) 6 [Pin it](#)



The photograph shows a desk setup. In the center is a yellow poster with the text 'IN HACK WE TRUST' and the word 'HACK' in large, 3D, blue and white letters. To the left is a yellow sticky note with a diagram and text. To the right is a green poster with various icons and text. The desk also has a 'UNITE' sticker and a 'facebook Hack' sticker.

Next Steps

- CS 6353 Compiler Construction
 - we learned how to design and analyze a language
 - 6353 teaches how to build a compiler for a language
- CS 6374 Computational Logic
 - learn about automated theorem proving
 - tools for doing formal software verification
- CS 6301-005 Language-based Security (shameless plug)
 - type theory for security analysis & enforcement
 - information flow, access control, etc.
- Independent study research
 - Dr. Gupta: logic programming
 - Me: language-based security

Course Evaluations

- Online
 - Please provide (constructive!) feedback
 - Non-anonymous feedback is even more helpful!
 - I never allow student comments (negative or positive) to affect the student's grade, so please don't worry about that.
- Some issues of interest...
 - Topics you wish had been included but weren't?
 - Homework/exam difficulty level
 - Helpfulness of instructor/TA
 - No textbook (make Winskel a required text?)

Questions?
Feedback?