Programming Languages Overview

Frank Pfenning

Carnegie Mellon University

Computer Science Open House

March 2004

PL Research at CMU

- The *Principles of Programming* (POP) group
  - Members
  - Research projects
  - Course offerings
- Project samplers (subjective selection!)
  - *ConCert*: Language Technology for Trustless Software Dissemination
  - *Triple*: Type Refinement in Programming Languages
  - *Twelf*: Logical and Meta-Logical Frameworks

Core Members

- *Stephen Brookes*, concurrency
- *Karl Crary*, certified code, typed compilation
- *Robert Harper*, certifying compilation, logical frameworks, module systems, type refinement
- *Peter Lee*, proof-carrying code, compilers
- *Frank Pfenning*, logical frameworks, automated theorem proving, type refinement
- *John Reynolds*, imperative programming, reasoning about low-level languages
- *Dana Scott* (retd. May'03)

POP and Software Engineering

- *Jonathan Aldrich* (ISRI), language techniques in software engineering
- *Ed Clarke*, hardware and software verification, model checking
- *David Garlan*, software architecture
- *Mike Reiter* (ECE/CS), security
- *Bill Scherlis* (ISRI), software dependability
- *Dawn Song* (ECE/CS), security
- *Jeannette Wing*, software specification and verification, security

POP and Theory

- *Peter Andrews* (Math), automated theorem proving, higher-order logic
- *Jeremy Avigad* (Phil), theory of computation
- *Steven Awodey* (Phil), category theory
- *Guy Blelloch*, algorithms, parallelism, scientific computation
- Multidisciplinary program
  Pure and Applied Logic
  (Computer Science, Philosophy, Mathematics)
  `http://www.cs.cmu.edu/~pal`

More Project Samplers

- *Claytronics* (Seth Goldstein, Todd Mowry, et al., see separate talk)
- *Self-Adjusting Computation* (Guy Blelloch, Bob Harper)
- *Separation Logic* (John Reynolds)
- *ArchJava* (Jonathan Aldrich)
- *Proof-Carrying Authorization* (Lujo Bauer, Frank Pfenning, Mike Reiter)
- *Model-Checking* (Ed Clarke, Jeannette Wing)

Some Recent Graduates

- Andrej Bauer, University of Ljubljana
- Lars Birkedal, IT University of Copenhagen
- Perry Cheng, IBM Research
- Alberto Momigliano (Phil), University of Edinburgh [postdoc]
- Robert OCallahan, IBM Research
- George Necula, Berkeley
- Gerald Penn (LTI), University of Toronto

More Recent Graduates

- Brigitte Pientka, McGill University
- Jeff Polakow, University of Edinburgh [postdoc]
- Carsten Schuermann, Yale
- Chris Stone, Harvey Mudd College
- Roberto Virga (Math), Princeton [postdoc]
- HongWei Xi (Math), Boston University

Some Course Offerings

- 15-714* Type Systems (Harper/Fa04)
- 15-812* Semantics (Brookes/S04)
- 15-819 Advanced Topics in PL
  - Type Refinement (Harper/Fa03)
  - Reasoning about Low-Level Languages (Reynolds/Sp02,Sp04)
  - Advanced Module Systems (Harper/Fa02)
  - Concurrency and Mobility (Harper/Sp02)

More Course Offerings

- 15-851 Computation and Deduction (Crary/F04)
- 15-815 Automated Theorem Proving (Pfenning/Sp04)
- 15-816 Linear Logic (Pfenning/Fa01)

Factors in Programmer Productivity

- Programmer productivity
  - Initial development time
  - Program correctness and robustness
  - Software maintainability
- Crucial factors
  - **Programming language(s)**
  - Development environment
  - Software engineering practices

Language Is Critical

- How do we implement data structures?
- How do we design and structure the code?
- How do we express assumptions and guarantees?
- How do we read and analyze a program?

Two Quotes

An ideal language allows us to express easily what is useful for the programming task and at the same time makes it difficult to write what leads to incomprehensible or incorrect programs.

—Nico Habermann

Good languages make it easier to establish, verify, and maintain the relationship between code and its properties.

—Robert Harper

Too Many Languages?

- In the last three years I have written code in at least the following languages:

Standard ML	Emacs Lisp	Twelf	PHP
TeX	Csh	C	MySQL
Perl	Java	CML	

- Different languages for different purposes
- Many are poorly designed
  - Those authors did not graduate from CMU!
  - Your favorite mis-feature?

Language Evaluation Criteria

- Some objective criteria
  - Is the grammar LALR(1)?
  - Is the language type-safe?
  - Is the language dynamically or statically typed?
  - Is the language Turing-complete?
  - Is the language call-by-value or call-by-name?
  - Is the language completely specified?
  - Does the language require a heap?
  - Does the language require dynamic dispatch?
- A subjective statement: “(I ((like Lisp)) (syntax))”

From the Perl Manual

When presented with something that might have several different interpretations, Perl uses the DWIM (that's "Do What I Mean") principle to pick the most probable interpretation. This strategy is so successful that Perl programmers often do not suspect the ambivalence of what they write. But from time to time, Perl's notions differ substantially from what the author honestly meant.

From the TEX manual

Please don't read this material until you've had plenty of experience with plain TEX. After you have read and understood the secrets below, you'll know all sort of devious combinations of TEX commands, and you will often be tempted to write inscrutable macros.

—Donald E. Knuth

Some Obfuscated TEX Code

```
\let~\catcode~‘76~‘A13~‘F1~‘j00~‘P2jdefA71F~‘7113jdefPALLF
PA’’FwPA;;FPAZZFLaLPA//71F71iPAHHFLPAzzFenPASSFthP;A$$FevP
A@@FfPARR717273F737271P;ADDFRgniPAWW71FPATTFvePA**FstRsamP
AGGFRruoPAqq71.72.F717271PAYY7172F727171PA??Fi*LmPA&&71jfi
Fjfi71PAVVFjbigskipRPWGAUU71727374 75,76Fjpar71727375Djifx
:76jelse&U76jfiPLAKK7172F71l7271PAXX71FVLnOSeL71SLRyadR@oL
RrhC?yLRurtKFeLPFovPgaTLtReRomL;PABB71 72,73:Fjif.73.jelse
B73:jfiXF71PU71 72,73:PWs;AMM71F71diPAJJFRdriPAQQFRsreLPAI
I71Fo71dPA!!FRgiePBt’el@ lTLqdrYmu.Q.,Ke;vz vzLqpip.Q.,tz;
;Lql.IrsZ.eap,qn.i. i.eLlMaesLdRcna,;!;h htLqm.MRasZ.ilk,%
s$;z zLqs’.ansZ.Ymi,/sx ;LYegseZRyal,@i;@ TLRlogdLrDsW,@;G
LcYlaDLbJsW,SWXJW ree @rzchLhzsW,;WERcesInW qt.’oL.Rtrul;e
doTsW,Wk;Rri@stW aHAHHFndZPpqar.tridgeLinZpe.LtYer.W,:jbye
```

Science of Programming Languages

- There is an established science of programming languages. Among its first papers:

  “Some Properties of Conversion”, Alonzo Church and J.B. Rosser, *Transactions of the American Mathematical Society*, Vol. 39(3), pp. 472–482, May 1936.

Basic Tools

- **Type theory**: Techniques for structuring languages to ensure safety and modularity of programs
- **Operational semantics**: Techniques for describing the execution behavior of programs, at various level of abstraction
- **Mathematical logic**: Techniques for specifying and verifying programs

ConCert

- *Language Technology for Trustless Software Dissemination* (NSF ITR)
- Faculty: Karl Crary, Robert Harper, Peter Lee, Frank Pfenning
- Students: Tom Murphy, Joe Vanderwaart, Leaf Petersen, Aleksey Kliger, Andrew Bernard, . . .
- Idea: reliable and secure grid computing via certified code
- `http://www.cs.cmu.edu/~concert`

Twelf

- *Logical and Meta-Logical Frameworks* (NSF GSC)
- Faculty: Frank Pfenning
- Students: Kevin Watkins, Jason Reed, Kaustuv Chaudhuri
- Idea: formal meta-language for specifying and reasoning about logics and programming languages
- `http://www.cs.cmu.edu/~twelf`

Triple

- *Type Refinement in Programming Languages* (NSF SEL)
- Faculty: Robert Harper, Frank Pfenning
- Students: Joshua Dunfield
- Idea: catching more programming errors at compile time via refined type analysis
- `http://www.cs.cmu.edu/~triple`

Summary: The POP Group

- is a close group of teachers and researchers with common values
- touches software engineering, formal methods, and theory
- proves theorems and builds systems
- welcomes new students
- has the best squash players on campus