Hack Your Language!

CS164: Introduction to Programming Languages and Compilers, Fall 2010

UC Berkeley upper-level elective course

Ras Bodik
Thibaud Hottelier
James Ide
The Why of Who and What

I used to teach a successful compiler course:

Aha! ugrads are future developers, not compiler jocks.

Also mindshare. Reverse the brain drain to AI and bio.
What’s one PL skill a developer should know?

Code reuse mechanisms have evolved:

- 1977 libraries of procedures
- 1986 design patterns
- 2006 DSLs

**DSL is how code is reused today.** Are students ready?

*Rinard:* Don’t write a DSL. Who will maintain it?

*Necula:* DSLs revolutionized a startup. But their programmers don’t dare to work with ASTs.
Take cs164. Become unoffshorable.

“We design them here, but the labor is cheaper in Hell.”
Principles behind course design

Teach PL principles via DSL design and implementation
   Hard in a messy language. DSLs in Lua, which is in Python.

Learn how to design and implement a DSL
   Final exam replaced with a project: create a DSL in 2 weeks.

Piggybacking: three birds with one stone
   coroutines + backtracking + regexes; ... 

Parsing: most of us want to deemphasize it
   Yet it’s one of the more useful skills. How to teach it?
Nine weekly projects

**interpreter** (abstractions)

**parser** (syntax-directed translation, external DSLs)

**web browser** (embedded DSLs, concurrency)
Language abstractions

prolog
with coroutines, it’s a few lines;
later used

yield: full coroutines
bytecode compiler, regex, backtracking

interpreter
closures, lexical scoping,
desugaring, iterators
The Parsing story

1. Write a random expression generator.

2. Invert this recursive generator into a parser by replacing print with scan and random with oracle.

3. Now write this parser in Prolog, which is your oracle. This is your recursive descent parser.

4. Look, this prolog parser has no negation. It’s Datalog!

5. Datalog can be evaluated bottom-up, with dynamic programming. Now we got CYK parser.

6. Datalog evaluation can be optimized with a Magic Set Transformation, which yields Earley Parser.
Parsing and external DSLs

**google calculator**

```
2/m/s
```

**syntax-directed translation**

disambiguation (%left, %prec)

**Earley parser**
Web browser: layout and scripting

Rx
from events to high-order dataflow; reactive programming a\'la RxJS

jQuery
embedded DSL for declarative DOM manipulation

layout engine
from constraints to attribute grammars; add OO to language
Contest winners in yellow jerseys
Testimonials

We are just about done with PA6, and I am still marveling at what we have created.

I think this class was an amazing first choice at an upper-div CS class. (Although the workload is pretty brutal)