

CIS 341: Compilers

Lecture 2

The Plan

- Today: Introduction to OCaml programming
 - A little background about ML
 - Interactive tour via the OCaml toplevel

- Reminder: Project 0 is available on the course web site.
 - Individual project – no groups
 - Due: 12 Sept. 2008 at 11:59pm
 - Advice: Start early! (This advice holds for *all* of the projects!)

ML's History

- 1971: Robin Milner starts the LCF (“logic of computable functions”) project at Stanford
- 1973: At Edinburgh, Milner implemented his theorem prover and dubbed it “Meta Language” – ML
- ML escaped into the wild and became “Standard ML” 1984
 - SML '97 newest version of the standard
 - There is a whole family of SML compilers:
 - SML/NJ – developed at AT&T Bell Labs
 - MLton – whole program, optimizing compiler
 - Poly/ML
 - Moscow ML
 - ML Kit compiler
 - MLj – SML to Java bytecode compiler
- ML 2000: failed revised standardization
- sML: successor ML – currently being discussed

OCaml's History

- The Formel project at the Institut National de Recherche en Informatique et en Automatique (INRIA)
- 1987: Guy Cousineau re-implemented a variant of ML
 - Implementation targeted the “Categorical Abstract Machine” (CAM)
 - As a pun, “CAM-ML” became “CAML”
- 1991: Xavier Leroy and Damien Doligez wrote Caml-light
 - Compiled CAML to a virtual machine with simple bytecode (much faster!)
- 1994: Development of CAML passed to project Cristal
- 1996: Xavier Leroy, Jérôme Vouillon, and Didier Rémy
 - Add an object system OCaml
 - Add native code compilation
- Many updates, extensions, since...
- Microsoft's F# language is a descendent of OCaml

OCaml Tools

- `ocaml` – the top-level interactive loop
 - `ocamlc` – the bytecode compiler
 - `ocamlopt` – the native code compiler
 - `ocamldep` – the dependency analyzer
 - `ocamldoc` – the documentation generator
 - `ocamllex` – the lexer generator
 - `ocamlyacc` – the parser generator
-
- We will also use `omake` (not part of the OCaml distribution)
 - A build system (like `make`) but written in and tailored for OCaml

Pervasive Types

- `int` -1, 0, 1, 2, etc. (note: not 32 bit!)
- `int32` 0l, 1l, 2l, etc.
- `int64` 0L, 1L, 2L, etc.
- `float` 0.0, 1.0, 2.0, etc.
- `string` "hello"
- `char` characters 'a', 'b', 'c', etc.
- `bool` true or false
- `unit` "trivial" type with only one value: ()
- `t * u * v` (1, "hello", 3.0) tuples – n-ary
- `t -> u` function from t to u
- 'a list [1;2;3] or ["a"; "b"; "c"] generic lists
- 'a option generic option types: None or Some v

Other Datatypes

- Records `{name="Ellie"; age=3}:{name:string; age:int}`
- 'a array `[|1;3;5|] : int array`
- 'a exn exceptions
- User-defined algebraic datatypes:
 `type foo = A | B of foo * int`
- Modules
- Objects

Libraries

- Generic container classes:
 - Lists, arrays, buffers, hash tables, queues, sets, etc.
- I/O, pretty printing
- Numeric libraries
 - Int32, Int64, bignums, etc.
- Lexing, parsing, streams, marshalling
- Regular expressions
- Unix interface
- Threads
- Graphics