OCaml on a JVM using OCaml-Java

Xavier Clerc — ocamljava@x9c.fr

OCamlMeetingParis 2008

26 January 2008

Outline

- Motivation
- Existing software
- Objectives
- Key points
- Subprojects
- Compatibility
- Roadmap

Motivation

	OCaml	Java
Language	**expressive**	verbose
Community	small	**huge**
Libraries	few	**many**
Code quality	**high**	inconsistent

Mixing allows to access the best of both worlds

Existing software

- JavaCaml http://www.ocaml-programming.de/javacaml

  interpreter of OCaml bytecode written in Java

- CamlJava http://pauillac.inria.fr/~xleroy/software.html#camljava

  OCaml / Java interface through JNI

- O'Jacare http://www.pps.jussieu.fr/~henry/ojacare

  interface generator for CamlJava

Objectives

- 100% pure Java - no JNI
- Both interpreted and compiled
- Easy access to Java classes
- No special runtime when compiling with **`ocamlc`**
- Compatibility with the original implementation
- Several OCaml programs running in the same JVM

Key points

- http://ocamljava.x9c.fr - ocamljava@x9c.fr
- Current version : 1.0 alpha (OCaml 3.10.0)
- Beta should be released in February (OCaml 3.10.1)
- Java 1.5
- Whole standard library (incl. lexing, parsing, marshalling)
- Libraries : bigarray, dbm, dynlink, graph, num, str, unix, threads
- Already able to run toplevel / to build a working `ocamlc.jar`

Subprojects

Barista	bytecode generation
Cadmium	interpreter & runtime support
Cafesterol	OCaml-to-Java compiler
Nickel	bindings generator
OCamlScripting	scripting engine for Java

Barista

- Library for class file manipulation
- Assembler / disassembler
- Implements the whole Java 1.5 specification
- Dependencies : Camlzip, Camomile
- Released under LGPL v3

Barista

```
.class public final pack.Test
.extends java.lang.Object

.method public static void main(java.lang.String[])
     getstatic java.lang.System.out : java.io.PrintStream
     ldc "hello."
     invokevirtual java.io.PrintStream.println(java.lang.String):void
     return
```

```
let instructions = [
    Instruction.GETSTATIC ((utf8_for_class "java.lang.System"),
                           (utf8_for_field "out"),
                           (`Class (utf8_for_class "java.io.PrintStream")));
    Instruction.LDC (`String (utf8 "hello."));
    Instruction.INVOKEVIRTUAL ((utf8_for_class "java.io.PrintStream"),
                               (utf8_for_method "println"),
                               ([(`Class (utf8_for_class "java.lang.String"))],
                                `Void));
    Instruction.RETURN
  ]
```

Cadmium

- Java port of `ocamlrun`
- Runtime support for Cafesterol-compiled programs
- Implements the whole OCaml bytecode instruction set
- Implements all primitives except the ones from labltk
- Dependencies : none
- Released under LGPL v3

Cadmium

```
@PrimitiveProvider
public final class Str {

   @Primitive
   public static Value caml_string_get(final CodeRunner ctxt,
                                       final Value s,
                                       final Value idx)
       throws Fail.Exception {
       final Block block = s.asBlock();
       final int i = idx.asLong();
       if ((i < 0) || (i >= block.sizeBytes())) {
           Fail.arrayBoundError();
       } // end if
       return Value.createFromLong(block.getUnsignedByte(i));
   }

}
```

Cafesterol

- Provides `ocamljava`, counterpart of `ocamlc` / `ocamlopt`
- Implements all language constructs
- Support standalone compilation or library sharing
- Dependencies : Camlzip, Barista, OCaml sources
- Released under QPL v1

Cafesterol

	ocamlc	ocamlopt	ocamljava
compiled interface	.cmi	.cmi	.cmi
compiled implementation	.cmo	.cmx	.cmj
implementation binary	-	.o	.jo
library	.cma	.cmxa	.cmja
library binary	-	.a, .so, ...	.jar

- Default is dynamic linking (Java style)
- "Standalone" linking is available (OCaml style)
- Can link as applet / servlet

Nickel

- Generates OCaml bindings for Java class
- Uses OCaml object system
- Supports callbacks
- Dependencies : none
- Released under GPL v3

Nickel

```xml
<?xml version="1.0" encoding="iso-8859-1"?>

<!DOCTYPE module SYSTEM "dtds/module.dtd">

<module name="Java">
  <interface java-name="java.awt.event.ActionListener"
             ocaml-name="jActionListener"
             wrapper="yes">
    <methods pattern="*(*)"/>
  </interface>
  <class java-name="javax.swing.JFrame" ocaml-name="jFrame">
    <constructor signature="(java.lang.String)"/>
    <method signature="getContentPane()"/>
    <method signature="setSize(int,int)"/>
    <method signature="setVisible(boolean)"/>
  </class>
</module>
```

Nickel

```
class jActionListener :
  [< `Cd'init of Cadmium.java_object
   | `Cd'initObj of < cd'this : Cadmium.java_object; .. >
   | `Cd'wrap of < actionPerformed : CadmiumObj.jObject -> unit; .. >
   | `Null ] ->
  object
    method actionPerformed : CadmiumObj.jObject -> unit
    method clone : CadmiumObj.jObject
    method equals : CadmiumObj.jObject -> bool
    method getClass : CadmiumObj.jClass
    method hashCode : int32
    method notify : unit
    method notifyAll : unit
    method toString : string
    method wait : unit
    method wait'1 : int64 -> unit
    method wait'2 : int64 -> int32 -> unit
  end
```

Nickel

```
class quit = object
  method actionPerformed (e : jObject) = exit 0
end

let () =
  let frame = new jFrame (`String "Nickel test") in
  let text = new jTextArea (`String ("This is Nickel/Cadmium")) in
  let view = new jScrollPane (`Component (text :> jComponent)) in
  let button = new jButton (`String "OK") in
  let listener = new jActionListener (`Cd'wrap (new quit)) in
  button#addActionListener listener;
  ignore (frame#getContentPane#add "Center" (view :> jComponent));
  ignore (frame#getContentPane#add "South" (button :>
jComponent));
  frame#setSize 320l 240l;
  frame#setVisible true
```

OCamlScripting

- Implements JSR 223 (`javax.script`)
- Supports script compilation
- Bindings can be defined
- Dependencies : Cadmium, Cafesterol
- Released under LGPL v3

OCamlScripting

```
final ScriptEngine engine = getEngine();
final ScriptContext ctxt = new OCamlContext(System.out,
                                            System.err,
                                            System.in);
ctxt.getBindings(ScriptContext.ENGINE_SCOPE).put("n", 5);
final String script =
    "let tmp_n : int32 = Cadmium.get_binding \"n\" in\n" +
    "let n = Int32.to_int tmp_n in\n" +
    "let a = Array.init n (fun i -> i * i) in\n" +
    "let sum = Array.fold_left (+) 0 a in\n" +
    "Printf.printf \"sum([0;%d[) = %d\\n\" n sum; sum\n";
final result = (((Value) engine.eval(script, ctxt)).asLong();
```

Compatibility (general)

- Big-endian / 32-bit implementation
- Unsafe features may behave differently (or even fail)
- Some Unix primitives are *emulated*
- Fonts are different (Graphics module)
- http://cadmium.x9c.fr/distrib/cadmium-compatibility.pdf

Compatibility (Cafesterol)

- Evaluation order
- Object cache not implemented
- Pending signals checked at given points
- Stack overflow / memory shortage not caught
- Rudimentary backtrace support
- Tail calls optimized only for direct recursion
- Very big (inlined) functions may fail to compile due to a Java constraint regarding maximum method size

Roadmap

- 1.0 alpha - september 2007
- 1.0 beta - february 2008
- 1.0 final - april 2008

- 1.x - work on compatibility, features, Java 1.6
- 2.x - work on performance issues
- 3.x - convergence to OCaml version number