

Clojure Compilation, Backwards

Nicola Mometto (Bronsa)

brobronsa@gmail.com

@Bronsa_

<https://github.com/Bronsa>

Clojure Compilation, Backwards

- Clojure compilation
- JVM bytecode
- tools.decompiler architecture overview
- demo

clojure compilation overview

- reader (text -> data)
- macroexpander (code -> code)
- analyzer (code -> AST)
- emitter (AST -> bytecode)
- AOT ns emitter (bytecode -> ns init classfile)

JVM bytecode

- Constant Pool (class/method refs, constant values)
- Class Fields
- Method Bodies

JVM bytecode

Method Bodies

- Local Variable Table
 - start, end offset, slot, name, signature
- Exception Table
 - start, end, catch handler, exception type
- Line Number Table
- Bytecode

tools.decompiler

- bytecode loader/parser (bytecode -> symbolic bytecode)
- analyzer (bytecode -> AST)
- sweetener (AST -> sugared AST)
- compiler (sugared AST -> code)
- (macro)compactor (code -> compacted code)
- pprinter (compacted code -> formatted text)

bytecode loader/parser

- BCEL to load bytecode
- parse class fields
- parse class methods
 - build exception table
 - build local variable table
 - build jump table
 - build symbolic bytecode vector

bytecode loader/parser

```
#:class{:name "test$foo",
  :super "clojure.lang.AFunction",
  ...
  :methods [#:method{:name "invokeStatic",
    :return-type "java.lang.Object",
    :arg-types [],
    :bytecode [#:insn{:name "ldc",
      :length 2,
      :pool-element #:insn{:target-value "hello"
        ..}
      :label 0}
    #:insn{:name "areturn", :length 1, :label 2}],
    :jump-table {0 0, 2 1}
    ..}
  ..]}
```


analyzer

- dispatches on super class/interface/class name
 - AFunction/RestFn -> function
 - IType/IRecord -> deftype/defrecord
 - IObj and name contains "reify" -> reify
 - name ends with "__init" -> namespace initializer

analyzer

- ignore abstract/bridge methods
- initialize static fields
- collect methods to decompile (invoke, load, deftype methods)
- process each method
 - stack machine interpreter to build AST
 - process using ctx map (statements, stack, pc, lvt, etc)

analyzer

- instruction types
 - stack ops (dup, swap, ..)
 - branch ops (ifeq, instanceof, ..) -> conditionals
 - local variable ops (aload, astore, ..) -> lexical blocks
 - math ops (ladd, ldiv, ..) -> math intrinsics
 - other ops (invokevirtual, athrow, putstatic, ..) -> all else

analyzer

```
(defmethod process-insn :dup [{:keys [stack] :as ctx} _]  
  (let [val (peek stack)]  
    (-> ctx  
      (update :stack conj val))))
```

;; simplified version

```
(defmethod process-insn :getstatic [ctx {:insn/keys [pool-element]}]  
  (let [{:insn/keys [target-class target-name]} pool-element]  
    (-> ctx  
      (update :stack conj {:op :static-field  
                           :target target-class  
                           :field target-name}))))
```

analyzer

```
(defmethod process-insn :ifeq [{:keys [stack] :as ctx} insn]
  (let [else-label (goto-label insn)
        goto-end-insn (insn-at ctx {:label else-label :offset -2})

        end-label (goto-label goto-end-insn)

        {then-label :insn/label} (insn-at ctx {:offset 1})

        test (peek stack)]

    (-> ctx
      (update :stack pop)
      (process-if test [then-label (:insn/label goto-end-insn)]
                    [else-label end-label]))))
```

analyzer

```
{:op :fn,  
 :name "foo",  
 :fn-methods [{:op :fn-method,  
                :fn-name "foo",  
                :var-args? false,  
                :args (),  
                :body {:op :do,  
                      :statements [],  
                      :ret {:op :const,  
                           :val "hello"}}}]}
```

sweetener

- pass over AST to add some syntax sugar
- some overlap with compactor
- e.g. `clojure.lang.PersistentVector/EMPTY` -> `[]`

(macro)compactor

- need to undo inlining/macroexpansion
- hardcoded patterns for clojure.core macros
- extension point with DSL for user-defined macros

(macro)compactor

attempt #1

- Kibit
- core.logic as a unification engine
- too slow
- no defined ordering is a problem

(macro)compactor

attempt #2

- core.match to pattern match over EDN
- fast!
- DSL over `match` to remove boilerplate
- no unification, use guards to enforce unification
- compact in postwalk over the source

(macro)compactor

attempt #2

(compact expr

```
[(`let [?t ?x]
  (if ?t ?y ?t))
```

```
{?t #(and (symbol? %)
  (-> % name (.startsWith "and__"))))}
```

```
:-> `(and ~?x ~?y)]
```

```
...)
```

(macro)compactor

down the rabbit hole

**java.io.IOException: File name too long, compiling:
(clojure/tools/decompiler/compact.clj:150:21)**

```
clojure.tools.decompiler.compact$macrocompact_step$fn__6117$fn__6118$fn__6119$fn__6120$fn__6121$fn__6122$fn__6123$fn__6124$fn__6125$fn__6126$fn__6127$fn__6128$fn__6129$fn__6130$fn__6131$fn__6132$fn__6133$fn__6134$fn__6135$fn__6136$fn__6137$fn__6138$fn__6139$fn__6140$fn__6141$fn__6142$fn__6143$fn__6144$fn__6145$fn__6166$fn__6167$fn__6168$fn__6195$fn__6200$fn__6201$fn__6202$fn__6203$fn__6204$fn__6205$fn__6206$fn__6207$fn__6208$fn__6209$fn__6210$fn__6232$fn__6233$fn__6234$fn__6235$fn__6238$fn__6241$fn__6244$fn__6247$fn__6250$fn__6251$fn__6254$fn__6255$fn__6256$fn__6257$fn__6258$fn__6262$fn__6266$fn__6269$fn__6270$fn__6271$fn__6274$fn__6278$fn__6279$fn__6280$fn__6284$fn__6285$fn__6286$fn__6287$fn__6300$fn__6301$fn__6302$fn__6303.invoke()
```

(macro)compactor

down the rabbit hole

- workaround patch to Compiler.java
- it worked!
- but one thread kept 100% CPU

(macro)compactor

down the rabbit hole

- BCEL
- core.match
- fipp

(macro)compactor

down the rabbit hole

- ~~BCEL~~
- core.match
- fipp

(macro)compactor

down the rabbit hole

- ~~BCEL~~
- ~~core.match~~
- fipp

(macro)compactor

down the rabbit hole

- ~~BCEL~~
- ~~core.match~~
- ~~fipp~~

(macro)compactor

down the rabbit hole

- ~~BCEL~~
- ~~core.match~~
- ~~fipp~~
- clojure compiler?

(macro)compactor

down the rabbit hole

- ~~BCEL~~
- ~~core.match~~
- ~~fipp~~
- ~~clojure compiler?~~

(macro)compactor

down the rabbit hole



(macro)compactor

down the rabbit hole

- ~~BCEL~~
- ~~core.match~~
- ~~fipp~~
- ~~clojure compiler?~~
- JVM?

(macro)compactor

down the rabbit hole

```
2546 Thread_11839119: Java: C1 CompilerThread3
+ 2546 thread_start (in libsystem_pthread.dylib) + 13 [0x7ffffde25708d]
+ 2546 _pthread_start (in libsystem_pthread.dylib) + 286 [0x7ffffde257887]
+ 2546 _pthread_body (in libsystem_pthread.dylib) + 180 [0x7ffffde25793b]
+ 2546 java_start(Thread*) (in libjvm.dylib) + 246 [0x10908a5b2]
+ 2546 JavaThread::run() (in libjvm.dylib) + 450 [0x10916c1fc]
+ 2546 JavaThread::thread_main_inner() (in libjvm.dylib) + 155 [0x10916ab0f]
+ 2546 CompileBroker::compiler_thread_loop() (in libjvm.dylib) + 657 [0x108db8d0f]
+ 2546 CompileBroker::invoke_compiler_on_method(CompileTask*) (in libjvm.dylib) + 1458 [0x108db67c8]
+ 2546 Compiler::compile_method(ciEnv*, ciMethod*, int) (in libjvm.dylib) + 144 [0x108ce8e9a]
+ 2546 Compilation::Compilation(AbstractCompiler*, ciEnv*, ciMethod*, int, BufferBlob*) (in libjvm.dylib) + 418 [0x108ce8718]
+ 2546 Compilation::compile_method() (in libjvm.dylib) + 109 [0x108ce8503]
+ 2546 Compilation::compile_java_method() (in libjvm.dylib) + 88 [0x108ce82da]
+ 2546 Compilation::build_hir() (in libjvm.dylib) + 280 [0x108ce81bc]
+ 2546 IR::compute_code() (in libjvm.dylib) + 43 [0x108cfd243]
+ 2546 ComputeLinearScanOrder::ComputeLinearScanOrder(Compilation*, BlockBegin*) (in libjvm.dylib) + 482 [0x108cfd1fc]
+ 2546 ComputeLinearScanOrder::compute_order(BlockBegin*) (in libjvm.dylib) + 441 [0x108cfd4d]
+ 2546 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [0x108cfc36b]
+ 2546 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [0x108cfc36b]
+ 2546 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [0x108cfc36b]
+ 2546 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [0x108cfc36b]
+ 2544 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [0x108cfc36b]
+ ! 2540 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [0x108cfc36b]
+ ! : 2525 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [...]
+ ! : | 2487 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [...]
+ ! : | + 2403 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [...]
+ ! : | + ! 2259 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [...]
+ ! : | + ! : 1975 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [...]
+ ! : | + ! : | 1592 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [...]
+ ! : | + ! : | + 1167 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [...]
+ ! : | + ! : | + ! 737 ComputeLinearScanOrder::compute_dominator(BlockBegin*, BlockBegin*) (in libjvm.dylib) + 121 [...]
+ ....
```

(macro)compactor

down the rabbit hole

- core.match is implemented as a transpiler
- wrote new backend, continuations instead of exceptions for backtracking
- no more C1 issues
- ~4x slower than original backend

(macro)compactor

down the rabbit hole

- alias no-op continuations
- compilation times skyrocketed
- coincidentally, compiler bug found just a week prior
- skip altogether no-op continuations

(macro)compact

light at the end of the tunnel



pprinter

- trim constant statements
- elide referred namespaces from symbols
- use aliases instead of long namespaces in symbols
- use fipp to pretty print the output in a readable format

DEMO

limitations

- locals are munged, symbols could collide
- no type hints/mutable deftype fields printing support
- casts/widening missing or extra ones inserted
- genclass/definterface not properly supported
- not all macros are compacted
- some extra garbage is produced

future work

- detect and sugar syntax-quote expressions
- compact for/doseq/ns
- properly support type-hints/mutable deftype field decls
- use line table to guide pprinting

Questions?

<https://github.com/Bronsa/tools.decompiler>