Compiling Scheme to C

Matt Might
University of Utah
www.ucombinator.org
matt.might.net

Administrivia

- Scheme to Java code posted
- Self-grading mechanism posted

Today

- Translate Scheme into C
- Mutable variable elimination
- Flat-env closure conversion

Strategy

- Create encoding of Scheme values in C
- Transform Scheme into subset of Scheme
- Map subset of Scheme into C program

Scheme values in C

- Word-tagged encoding
- Bit-tagged encoding
- Tag-vector encoding

Word-tagged encoding

- Attach a tag to every value with its type
- Use enum {} to create a set of tag values

Example tags

- VOID
- INT
- BOOLEAN
- CLOSURE
- CELL
- ENV
- OBJECT

Bit-tagged encoding

- Reserve part of each word for tag
- Alignment hack tag bits for pointers
- Other word types lose precision

Example

- 64-bit words = 8-byte aligment = 3 tag bits

Tag-vector encoding

- Keep tags and values in separate areas
- Extra argument to procs: tag vector
- Extra local variable: local tag vector

scheme.h

Core language

```
<exp> ::= <const>
        |  <prim>
        |  <var>
        |  (if <exp> <exp> <exp>)
        |  (set! <var> <exp>)
        |  (lambda (<var> ...) <exp>)
        |  (<exp> <exp> ...)

<const> ::= <int>
          |  #f
```

Core language

```
<exp> ::+ (let ((<var> <exp>) ...) <exp>)
        |  (letrec ((<var> <exp>) ...) <exp>)
        |  (begin <exp> ...)
```

IR (I)

```
<exp> ::+ (cell <exp>)
        |  (cell-get <exp>)
        |  (set-cell! <exp> <value>)
```

IR (2)

```
<exp> ::+ (closure <lambda-exp> <env-exp>)
        |   (env-make <env-num>
                      (<symbol> <exp>) ...)
        |   (env-get <env-num> <symbol> <exp>)
```

Closure conversion

```
(lambda_k (v_1 ... v_n)
    ... x_i ...)
```

=>

```
(closure (lambda_k ($env v_1 ... v_n)
           ... (env-get k x_i $env) ...)
         (make-env k (x_1 x_1)... (x_m x_m)))
```

($x_1, \ldots, x_m$ are free variables)

Closure conversion

(f e_1 ... e_n)

=>

((fun->proc f) (fun->env f) e1 ... en)