2008 年 2 月

- 前の月: Memo200801
- 次の月: Memo200803

2008-02-13 (wed)

ここ[*1]の、Shiro さんコメントが面白いという噂。ほんとだ。

iscm003

version 0.0.x 系列は機能追加はせずに、リファクタリングとバグとりと、現在の機能でどのへんに限界があるかのお勉強……とか思っていたんだけど、飽きた（笑）。
要するに、さっさとマクロを実装してみたくなったのだ。
というわけで、0.0.x 系は、ひとまずおしまい。

- iscm003.tar.bz2[*2]

これにて、0.1.x 系に移行します。あんまり「計画性」とかに拘らず、気が向いたところに手をつけていくことにします。

- (define (f n) ...) タイプの関数定義
- lambda の仮引数形式で未サポートなのをなくす
- 健全なマクロ
- integer 以外のデータタイプ

これらを実装したら、ぼちぼちとライブラリを書いていこうと思う。
また、「行数をなるべく増やさないでおきたい則」は解除。ソースを小さくすることより、使いやすいプログラムにすることを重視しましょう。

いよいよマクロ

ああ、体調わる。頭いてえよ。
だからってわけでもないんだろうけど、パターン言語の仕様がいまいちわかんないよ。うー。
とりあえず、以下~~ふたつ~~みっつを実現することをめざす。

```
(define-syntax let
  (syntax-rules ()
    ((let ((name val) ...) body1 body2 ...)
     ((lambda (name ...) body1 body2 ...)
```

[*1] http://d.hatena.ne.jp/archacker/20080203/1201988205
[*2] http://uhideyuki.sakura.ne.jp/files2008/iscm003.tar.bz2

```
       val ...))))

(define-syntax let*
  (syntax-rules ()
    ((let* () body1 body2 ...)
     (let () body1 body2 ...))
    ((let* ((name1 val1) (name2 val2) ...)
       body1 body2 ...)
     (let ((name1 val1))
       (let* ((name2 val2) ...)
         body1 body2 ...)))))

(define-syntax begin
  (syntax-rules ()
    ((begin exp ...)
     ((lambda () exp  ...)))))
```

このみっつで、実用的なパターンは出尽くしているんではないか……と思っている。

2008-02-12 (tue)

hygienic macro

以前 Scheme:マクロ:CommonLisp との比較:意味論[*3]（と、一連の記事）を読んだときには、あんまり内容をフォローできなくて、「黒田さんって恐そうな人だなぁ」くらいの感想しか抱けなかったんだけど、いま読むと少しわかる気がする。

iscm1 へインプリするマクロは、はじめから hygienic macro をめざそう。

自由変数の衝突 (Scheme:マクロ:CommonLisp との比較)[*4]にかいてある、「R5RS マクロでは、マクロの定義環境での意味がそのまま保持されます。」ってのは、一瞬どっきりしたけど……

ちょっと実験：

```
(let ((< (lambda args #f)))
   (arithmetic-if -1 'neg 'zero 'pos)) => neg
```

これは、例にある通り。それじゃあ、ってんで、次のようにやるとどうなるか。

```
(set! < (lambda args #f))
(arithmetic-if -1 'neg 'zero 'pos) => pos
```

ほっ。やっぱりそうか。lambda でレキシカル・スコープを実現したのと、同じことね。これなら実装できそうだ。

[*3] http://practical-scheme.net/wiliki/wiliki.cgi?Scheme:**マクロ**:CommonLisp **との比較**:**意味論**

[*4] http://practical-scheme.net/wiliki/wiliki.cgi?Scheme:**マクロ**:CommonLisp **との比較**#H-14z5rfn

束縛変数の衝突への対処は、処理系内部でしか使えないような名前（括弧を含むとか）に変換してやればいいのではないかと思っているんだが…。

2008-02-08 (fri)

いやぁ、自分で書いた scheme 処理系であそんだり、眺めたりするのは、楽しい！想像していた通りだ。

calla[*5]でなんとなく彷彿するものといえば、灰色の魔女[*6]？

iscm

やっぱり、「『とりあえずのでっち上げ』でもいいから」と思いつつ、とにかく動く処理系を書いてみる[*7]のは、正解だった。

楽しいし、いままで読んでもあまり頭に入らなかった R5RS の文章が、とてもよくわかる。

このへんで、iscm の今後の改造方針について、ちょっとメモっておこう。

iscm0 という名前の間は、Version 0.x.x となる。つまり最初の番号が 0 で、それはプログラム名 iscm? のとこにも入る。

Version 0.0.x 系列では、基本的に、機能追加はしない。

機能の不足や、僕自身が Scheme について誤解していたことによる変な仕様は直さない。これは、僕の現時点の理解をただしく写しているはずのものだ。

よって、ソースをきれいにしたり、プログラムミスによるバグの除去のみとする。いっそ、Version 0.0.x は、「C で 500 行」に（無理せず）近づけるのを目指してもいいかも[*8]。できるだけシンプルな処理系。

これらの作業中に、Version 0.1.x 系列にいれたい機能を考えておこう。

いま、思っていることのメモ：

lambda の仮引数部の形式は、R5RS (ないし R6RS) に書かれている全ての形式をサポートしたい。可変長引数を。

let がほしい。現状の cond のように built-in にしてしまえば、とにかく動く程度のものはすぐに作れそうだが、そうはしない。R5RS 7.3 に書いてあるとおりのマクロで実現する。つまり、iscm そのものに追加する機能は、「マクロ」。

ただし、Version 0.1.x 時点では、保健的であることを**要件とはしない**。まず、「let をうごかすにはこんなもんかな」という程度のものをつくってみて、その後、それではどう困るのかを確かめていく方向で。

さくっと保健的なのが作れそうなら、それでもいいけど。

cat lib4iscm0.scm hoge.scm | iscm0 とかやんなくても、ライブラリは勝手に読み込んでほしい。また、それに類する使い勝手の改良も。

めも

typo: langage -> language

[*5] http://uhideyuki.sakura.ne.jp/uikitexi/index.cgi?Memo200801#p2008012410

[*6] http://www.google.co.jp/search?q=**灰色の魔女**

[*7] http://uhideyuki.sakura.ne.jp/uDiary/?date=20080207#p01

[*8] 空行をカウントしないことにすれば、なんとかいけそう。

命名の不徹底。十年くらい前に罹患した、命名規則病（細目：ハンガリー症候群）の後遺症がちらほら。
p とか付けない方向で統一しよう。
const 修飾は、ついてたり、ついてなかったりなので、きちんとつける。一応。

2008-02-05 (tue)

highslide.js

- Highslide JS のサンプル[*9]
- 配布元[*10]

かっちょいいな。

[*9] http://caramel-tea.com/sample_page/highslide.html
[*10] http://vikjavev.no/highslide/