

里々とLispとリーダーマクロ

zick(λ組)

うかべん 大阪 #4



zickって誰？

- 京都の方の大学生
- 大学に入ってからLispを知る
- これまで7つの言語でLisp処理系を書いた
 - C, JavaScript, NScripter, PostScript, PHP, Prolog, ニコスクリプト(ニワン語)
 - でも、どれもテキトーな作り。
粗製濫造が基本
- 『リリカルLisp』の中の人
- 『ニコ動でLisp』の中の人



役に立たないものを作るのが大好き

リリカルLisp

- (恐らく)世界初(そして最後)のLisp学習ゲーム
- Lispを勉強した「気分」になれる



ニコ動でLisp

- ニコニコ動画上で動作するLispインタプリタ
- 夢のお告げがきっかけでつくること

状態: END PRINTING 5 (< 1 2) car
処理: (* 10 (cos (/ (* 0 3.14159) 18
入力: (cdr '(a b c)) (1+2)
出力: (b c) 33

ただし、アトムは英字小文字のみである必要があります。たとえば、ichigo や kururu は使用できますが、LISP や r6rs は使用できません。

再生時	コメント
01:14	(+1 1)
01:22	(+ 100 500)
00:22	1+1
00:36	+
00:48	3+3
01:07	24
00:31	car
00:21	(+1 2)
01:14	10+2
00:35	(+1 1)
01:22	(+23)
01:47	(1+)
01:55	(1+)
00:16	(cdr '(a b c))

うかべん
大阪 #4

ニコ

本日のテーマその1

- 里々とLispは似ているのか？
 - Lispは汎用のプログラミング言語(*1)
 - 里々は単一目的のスクリプト言語
 - 共通点は「括弧」が沢山出てくること

- (*1)
「Lisp」は複数のプログラミング言語の総称
その言語を作った人が「Lisp」と名乗ればLispの仲間入り？
「Common Lisp」や「Scheme」のように汎用のものもあれば、
「elisp」や「AutoLisp」のように単一目的のものもある
ここでは主に「Common Lisp」を取り上げる



うかべん
大阪 #4

里々の場合(1)

- *夕方
: (季節) らしくなってきたね
: ((季節) の食べ物) が食べたい

サンマが食べたい!

- @季節

(季節)

=> 「春」「夏」「秋」「冬」のいずれかになる
(面倒なので必ず「秋」が出ることにします)

(秋の食べ物)

=> 「サンマ」「焼き芋」のいずれかになる

春夏秋冬

- @秋の食べ物

サンマ
焼き芋

秋らしくなってきたね



Lispの場合(1)

- (defun choice (lst)
 (nth (random (length lst))
 lst))
- (defun 季節 ()
 (choice '("春" "夏" "秋" "冬")))
- (defun 食べ物of (季節)
 (if (string= 季節 "秋")
 (choise '("サンマ" "焼き芋"))))

• (食べ物of (季節))
(季節)

=> 「春」「夏」「秋」「冬」のいずれかになる
(食べ物of "秋")

=> 「サンマ」「焼き芋」のいずれかになる



比較(1)

- 括弧の処理(*2)
 - 里々は単語群からの選択
 - Lispは関数呼び出し
- 入れ子の括弧
 - 両者とも内側から評価される(*3)
 - 里々は参照する単語の名前全体に影響する
 - Lispは引数として分離されてしまう
- (*2)
両者とも他の目的にも使用する
- (*3)
後述の通りLispは実は違う



Lispで無理やり里々の真似(1)

- ```
(defun 秋の食べ物 ()
 (choice ' ("サンマ" "焼き芋"))))
```
- ```
(defmacro like-satori (form)  
  (let* ((cells (remove-if-not #'consp form))  
         (vars (mapcar #'(lambda (_) (gensym)) cells)))  
    `(let , (mapcar  
            #'(lambda (v c) (list v (list 'like-satori c)))  
            vars cells)  
      (funcall  
        (symbol-function  
          (intern  
            (concatenate  
              'string  
              ,@ (mapcar  
                #'(lambda (e)  
                  (if (consp e)  
                    (pop vars)  
                    (symbol-name e))))  
              form))))))))))
```
- ```
(like-satori ((季節)の食べ物))
;=> 「サンマ」か「焼き芋」
```



## 里々の場合(2)

- (if, 条件, 真のときに返す値, 偽のときに返す値)
- (set, 変数名, 設定する値)
- 引数の区切りには「,」の他に「、」等も使える
- @誤った例  
(if, (乱数0~1) , (set, 値, 0) , (set, 値, 1) )
- @正しい例  
( (if、 (乱数0~1) 、 set, 値, 0、 set, 値, 1) )

括弧の中は「必ず」内側から評価されるので、  
一つ目の例は変数「値」に必ず1が代入される

内側の括弧 => “set, 値, 0”という文字列  
(set, 値, 0) => 変数「値」に0が代入



# Lispの場合(2)

- (if 条件 真のときに評価する式 偽のときに評価する式)
- (setq 変数名 設定する値)
- (if (/= (random 2) 0)  
 (setq 値 0)  
 (setq 値 1))

「関数呼び出し」では引数が先に評価される

しかし、ifやsetqは関数ではなく特殊オペレータ  
特殊オペレータやマクロの呼び出しはで引数は  
評価されない(されるとは限らない)



うかべん  
大阪 #4

# 比較(2)

- 引数の区切り
  - 里々では「、」や「,」を使うが、1つの括弧の中で最初に使った種類のみ有効で、他の種類が後で出ても無視
  - Lispでは空白や区切り文字(括弧など)で区切られる
- 引数の評価
  - 里々では必ず先に評価される
  - Lispでは特殊オペレータ等では評価されない場合がある



# Lispで無理やり里々の真似(2)

- 誤った例の真似

- (defun my-if (test then else)  
 (if test then else))
- (my-if (/= (random 2) 0)  
 (setq 値 0)  
 (setq 値 1))

- 正しい例の真似

- ごめん無理
- 名前の連結との共存が非常に面倒
- マクロによる展開は思いつかなかったけど、  
 インタプリタ的な動作でならなんとか...



# 余談: 先行評価とif

- 関数を呼ぶことにより回避

- 里々

```
((if, (乱数 0 ~ 1) , 処理1, 処理2))
```

```
@処理1
```

```
(set, x, 0)
```

```
@処理2
```

```
(set, x, 1)
```

- Lisp

```
(funcall
```

```
(my-if (random 2)
```

```
#' (lambda () (setq x 0))
```

```
#' (lambda () (setq x 1))))
```

里々にもlambdaがあればいいのに...



# テーマその1 まとめ

- 里々とLispは似ているのか？
  - 括弧を入れ子に出来るなど、構文は似ている
  - 評価の規則が大きく異なっている
  - 両者が「似てる」と一言で済ますのは少々問題があるかも



# 本日のテーマその2

- 構文が似てるなら里々のプログラムを無理やりLispのプログラムとして読めないか？
  - Common Lispにはリーダマクロという武器がある
  - Lispのプログラムとして読み込む = コンパイルできる  
=> コンパイラを作ったのと同じだよ

例えば計算式を読み込むリーダマクロを作る

```
$ x = 1 + 2 * 3
```

↓ (read : リーダマクロがLispの通常の式にする)

```
(setq x (+ 1 (* 2 3)))
```

↓ (compile : Lisp処理系が頑張る)

バイトコードや機械語に





# 里々からLispへの変形

- #元となる里々プログラム
  - \*起動
    - :おはー
    - :らっきー
- ;リードされたLispプログラム  
(deftalk 起動
  - (:change-talker)
  - "おはー"
  - (:change-talker)
  - "らっきー")
- ;マクロ展開後

```
(push
 (cons nil
 (lambda ()
 (with-output-to-string (s)
 (let ((*standard-output* s))
 (format t "\\0\\s[0]\\1\\s[10]")
 (format t "~a" "\\0")
 (format t "~a" "おはー")
 (format t "~a" "\\1")
 (format t "~a" "らっきー")
 (format t "\\e") s))))
 (gethash (symbol-name '起動) *talk-table*)))
```



# SHIORIとして動作させる(1)

- 本当はCLでDLLを作りたかったんだお



- けど、それが出来るECLは日本語に弱いんだお(\*1)



- だから、DLLはCで書いてパイプ通信するお！

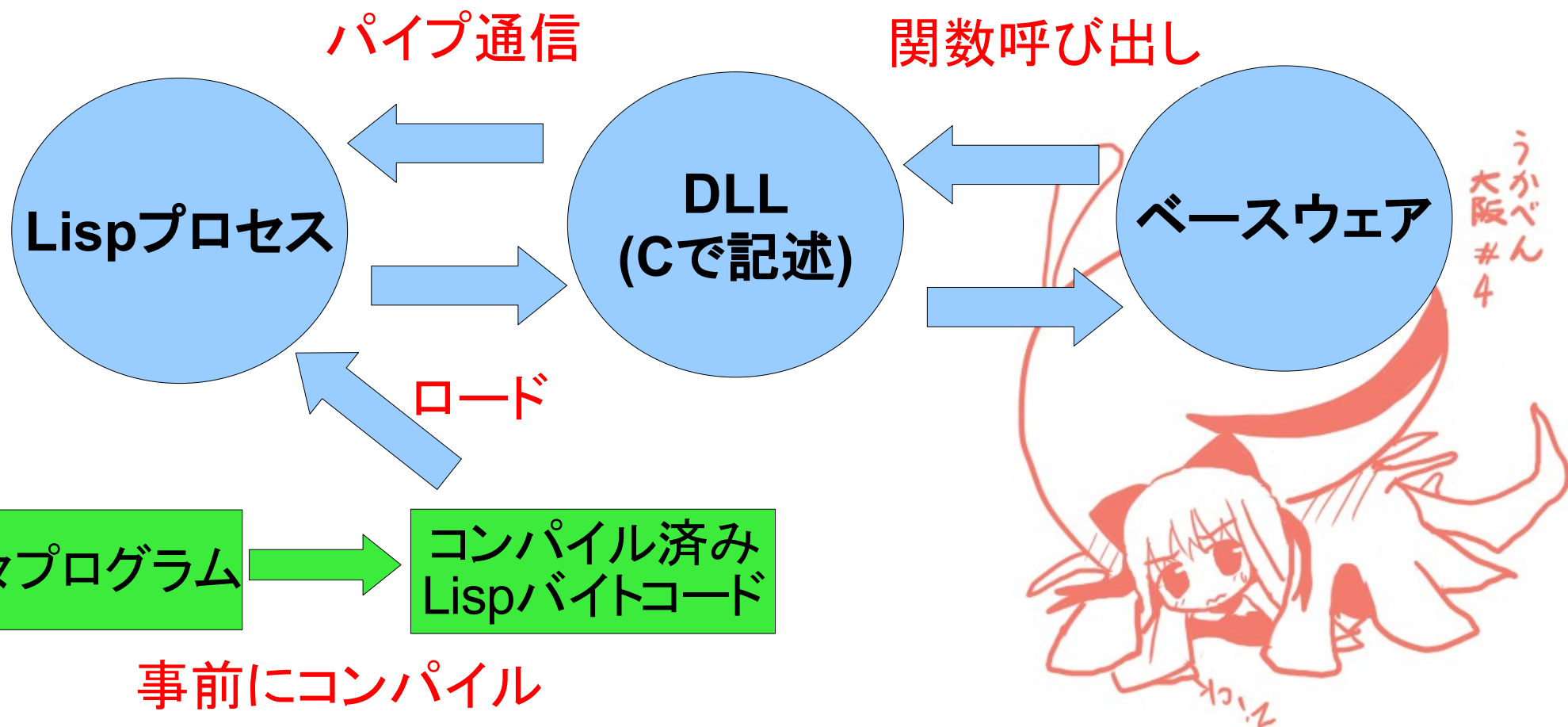


(\*1)商用の処理系なら他にも出来るものがあります



# SHIORIとして動作させる(2)

- Lispは標準入出力を使うだけで済む
- DLLは情報を左右に流すだけ



# 実演

- \* 起動

- : (0) Lispパワーで動いています

- : (1 0) といっても、見た目ではなににも分からないけど。

- : 実際に動作してる所を見せるといっても、特に何も出来ないんですよ。

- : とりあえず何かやれば？

- : それじゃあ、ずっとウマウマしておきますね。 (1 0 0 0)

- \* 終了

- : (0) えんいー

- : えんいー——！！！！



# テーマその2 まとめ

- 構文が似てるなら里々のプログラムを無理やりLispのプログラムとして読めないか？
  - 里々のサブセット読み込ませることは成功
    - こっちで動くものは本物の里々でも動く
  - 可能な限り、事前に単純なプログラムに展開
    - 単純な括弧や計算式は事前にコンパイル
    - 高速に動作する...気がします。気がするだけかもしれません
  - 複雑な括弧はインタプリタ的な動作
    - 文字列をリードしなおすため遅い
    - 単純な形でもトーク/単語/変数どの参照か分からない



# 実際に計測してみた(1)

- 目的はプログラムの特定の箇所の計測
- 事前に高速化できるものと出来ないものを計測
  - 単純な計算を行うもの (完全にコンパイルされる)
  - 複雑な構造の括弧を持つもの (インタプリタ的動作)
- Lisp処理系にはCLISP 2.45を使用
- 里々は適当に転がっていたもの。バージョン不明
- DLLのrequest関数の呼び出し時間を計測
- 計測に使ったマシン
  - OS : Windows XP
  - CPU: Celeron M 1.5GHz



# 実際に計測してみた(2)

- 計測方法(詳細は付録を参照)

- プログラム

- \$ 計算結果 =  $1 + 2 * 3 - 4 \div 2 + (\text{計算結果}) \% 5$

- この式が200回あるものと400回あるものを計測

- (400回の結果)-(200回の結果)で計算時間を求める

- 計測結果

- 里々 : 37000usec

- Lisp (未コンパイル) : 3400usec

- Lisp (コンパイル済) : 800usec

- 圧倒的な勝利!!!

- 里々の計算式の処理はかなり遅い？



# 実際に計測してみた(3)

- 計測方法(詳細は付録を参照)
  - プログラム  
: ( (季節) の食べ物) が食べたい!!!  
この行が50回あるものと100回あるものを計測
  - (100回の結果)-(50回の結果)で計算時間を求める
- 計測結果
  - 里々 : 7600usec
  - Lisp (未コンパイル) : 61800usec
  - Lisp (コンパイル済) : 16300usec
- 圧倒的な敗北!!!
- メモリ無駄使いしすぎ。GCがかなりの時間を食ってる
- 評価のアルゴリズムを直したらまだまだ速くなるかも





# 全体のまとめ

- 里々とLispは構文はある程度似ている
  - 評価規則は全く異なる
  - 里々にもlambdaとか先行評価でないものが欲しい
- CLで里々のプログラムを無理やり読ませた
  - コンパイル出来る部分は圧倒的に速い
  - それ以外の箇所はまだまだ工夫が必要
- で、それって役に立つの？
  - 立ちません

えんいー



# 参考文献

- レゴキチ/里々まとめ - 駄でべろぱの小ネタWiki  
<http://emily.shillest.net/specwiki/index.php?FrontPage>
- 電気で動くうにゅう・廃屋の夏  
<http://www.geocities.jp/poskoma/>
- CROW・SSPリファレンス  
<http://crow.aqrs.jp/reference/all/>



# 付録(1)

- 計測(1)に使ったプログラム

```
* calc
```

```
$ 計算結果 = 0
```

```
$ 計算結果 = 1 + 2 * 3 - 4 ÷ 2 + (計算結果) % 5
```

```
$ 計算結果 = 1 + 2 * 3 - 4 ÷ 2 + (計算結果) % 5
```

```
... (中略) ...
```

```
$ 計算結果 = 1 + 2 * 3 - 4 ÷ 2 + (計算結果) % 5
```

```
: anser = (計算結果) !
```

```
: What?
```



# 付録(2)

## ● 計測(2)に使ったプログラム

\* kakko

: ( (季節) の食べ物) が食べたい!!!

... (中略) ...

: ( (季節) の食べ物) が食べたい!!!

@季節

春

夏

秋

冬

@春の食べ物

つくし

竹の子

@夏の食べ物

スイカ

カキ氷

@秋の食べ物

サンマ

焼き芋

@冬の食べ物

おもち

みかん

