Lispkurs

Zum Praktikum "KI-basierte Robotersteuerung"

Lorenz Mösenlechner
moesenle@in.tum.de

Thomas Rühr
ruehr@in.tum.de

Macros

Lisp Programmevaluation

- Laufzeit (Funktionen und Methoden)
- Compilezeit (Macros)
- Parse-Zeit (Reader-Macros)

Macros

Lisp Programmevaluation

- Laufzeit (Funktionen und Methoden)
- Compilezeit (Macros)
- Parse-Zeit (Reader-Macros)

Was sind Macros?

- Lisp-Funktionen, die Code als Liste zurückliefern.
- Transformieren Code-Bäume in neuen Code

$\Rightarrow$ Programmierbare Programmiersprache

Beispiele

- Viele Lisp-Konstrukte, z.B. *when*, *cond*, *and*.
- Common Lisp Object System (CLOS)

Beispiele

- Viele Lisp-Konstrukte, z.B. *when*, *cond*, *and*.
- Common Lisp Object System (CLOS)

```
(defmacro when (condition &rest body)
  `(if ,condition
    (progn ,@body)))
```

Capturing

```
(defmacro test-capture (&rest body)
  `(let ( (tmp 'tmp) )
     ,@body))

(let ( (tmp 'xy) )
  (test-capture
    (format t "~a ~%" tmp)))
⇒ 'tmp
```

Capturing

```
(defmacro test-capture (&rest body)
  (let ( (tmp (gensym)) )
    `(let ( (,tmp 'tmp) )
       ,@body)))

(let ( (tmp 'xy) )
  (test-capture
    (format t "~a ~%" tmp)))
⇒ 'xy
```

with-gensyms

In komplexen Macros kommt es häuffig vor, das viele Symbole gebraucht werden.

```
(defmacro with-gensyms (syms &rest body)
  `(let ,(mapcar #'(lambda (sym) `(,sym (gensym))) syms)
    ,@body))
```

Wiederholung: Pattern-Matching

- *multiple-value-bind*
  Matching nur auf Listen, keine echten Patterns
- *match*
  Mächtiges Pattern-Matching, aber Bindings werden nur als Liste zurckgegeben.

$\Rightarrow$ Macro, um beides zu verbinden.

pat-match

```
(defmacro pat-match (pat-1 pat-2 &rest body)
  (with-gensyms (bdgs found?)
    `(multiple-value-bind (,bdgs ,found?) (match ',pat-1 ,pat-2)
       (when ,found?
         (let ,(mapcar #'(lambda (v) `(,v (binding ',v ,bdgs)))
                       (remove-if-not #'varsym? pat-1))
           ,@body)))))

(pat-match (test ?var-1 test2 ?var-2) '(test 1 test2 2)
  (format t "~a ~a~%" ?var-1 ?var-2))
```

strict-pat-match

```
(defmacro strict-pat-match (pat-1 pat-2 &rest body)
  (with-gensyms (bdgs found?)
    `(multiple-value-bind (,bdgs ,found?) (match ',pat-1 ,pat-2)
      (unless ,found?
        (error 'simple-error :format-control "Pattern does not match"))
      (let ,(mapcar #'(lambda (v) `(,v (binding ',v ,bdgs)))
                    (remove-if-not #'varsym? pat-1))
      ,@body))))
```

def-match-fun

```
(defmacro def-match-fun (name arg-desc &rest body)
  (with-gensym (args)
    `(defun ,name (&rest ,args)
       (strict-pat-match ,arg-desc ,args ,@body))))

(def-match-fun test-fn (args ?x ?y)
  (format t "~a ~a~%" ?x ?y))

(test-fn 'args ?x ?y)
```