In: Wittig, Th. (ed): LISP 1.6: Eine Erweiterung der Stanford-Version,
Hamburg: FB-Informatik, M-19/75, S. 7 -11.

4.3. Die Libraryfunktionen

4.3.1. Pattern Matching und die Transformation von S-Expressions

von Wolfgang Wahlster

1. Mit dem Package #PATTERN steht dem Benutzer vom LISP 1.6 II ein komfortabler Pattern Matcher zur Verfuegung. Pattern Matching und davon abhaengige Operationen haben sich als nuetzliche Werkzeuge innerhalb der Forschung zur Artificial Intelligence und Computational Linguistics erwiesen und sind daher meist integraler Bestandteil neuerer "very high level programming languages" [4] wie PLANNER, CONNIVER, QLISP, PLASMA und FUZZY.

#PATTERN besteht aus 45 von dem UCI LISP System [5] auf unser System portierten, teilweise korrigierten oder redefinierten Prozeduren.

2. Das Pattern Matching wird von der zweistelligen Praedikatsfunktion MATCHP geleistet. MATCHP prueft, ob das erste Argument (genannt: Struktur) ein Element der durch das zweite Argument (genannt. Pattern) beschriebenen Sprache ist.

```
                                          ---
                                         ¦ T falls Struktur ∈ L(PATTERN)
MATCHP(STRUKTUR,PATTERN)------->         <
                                         ¦ NIL sonst
                                          ---
```

Dabei wird eine Sprache durch Angabe einer endlichen Zahl von Eigenschaften, die nur Woerter dieser Sprache erfuellen, beschrieben. Auf diese Weise koennen durch Verwendung rationaler Ausdruecke innerhalb des Patterns Typ-3-Sprachen beschrieben werden. Indizierungen mit dem Postfix LABEL und "semantische" Tests ermoeglichen die Beschreibung von Klammerstrukturen, wie sie fuer Typ-2-Sprachen bekannt sind.

3. Die Funktion TRANSFORM leistet eine Abbildung eines symbolischen Ausdrucks S auf einen symbolischen Ausdruck S' in Abhaengigkeit von dem erfolgreichen Match des Patterns mit S.

```
                                          ---
TRANSFORM                                ¦ Ergebnis der Struk-
      (STRUKTUR                          ¦ -turaenderung falls
            (PATTERN STRUKTURAENDERUNG)) ---> <(MATCHP (...)
                                          ¦      = T
                                          ¦ NIL sonst
                                           ---
```

Nach der Terminologie der von N. Chomsky entwickelten Transformations-grammatik sind die beiden ersten Argumente von TRANSFORM die "structural description" und das dritte Argument (hier "Struktur-veraenderung" genannt) der "structural change". Wie Ritchie und Peters bewiesen haben, sind solche Transformationsgrammatiken mit kontextfreier Basis Typ-0-Grammatiken schwach aequivalent.

4.1. Als Pattern kann ein atomares Pattern oder eine Liste, deren Elemente Patterns sind, dienen. Ein LISP-Atom als einfachstes atomares Pattern matcht Strukturbaeume, in denen es die Wurzel ist.
Beispiele:
(MATCHP @ATOM @ATOM) = T
(MATCHP @(WURZEL (B C)) @WURZEL) = T

Listen von Patterns matchen genau dann mit Listen, wenn die entsprechenden Elemente der Listen matchen.
Beispiele:
```
(MATCHP @(DEC 10) @(DEC 10)) = T
(MATCHP @(A (B (C D)) E) @(A B E)) = T
```

Punktpaare matchen genau dann, wenn die entsprechenden Konstituenten des Punktpaares matchen.
Beispiele:
```
(MATCHP @(A.B) @(A.B)) = T
(MATCHP @(A.B) @(A.C)) = NIL
```

Zur Formulierung von Patterns sind ausserdem folgende terminale Zeichen vorgesehen: VAR,T,OPTIONAL,SEGMENT,STAR,OR,NOT,AND,PATTERN,LABEL,FUNCTION

4.2. Wird innerhalb eines Patterns ein Ausdruck mit dem Praefix VAR versehen, so wird er vor dem Matching-Prozess ausgewertet. Das Ergebnis der Auswertung wird an der entsprechenden Stelle des Patterns eingesetzt.
Beispiele:
```
(MATCHP @(2 HAMBURG) @((VAR (LENGTH @(A B))) HAMBURG)) = T
(MATCHP @(A (B C)) @(A (VAR(CDR @(A B C))))) = T
(MATCHP @((I F I)) @((VAR (EXPLODE @IFI)))) = T
```

4.3. Das terminale Zeichen T dient als atomares Pattern, das beliebige Atome matcht. Kommt der Matching-Prozess an eine Stelle, an der T in der CDR-Position steht, so wird jede von NIL verschiedene Restliste gematcht (Anwendung: Repraesentation von beliebig erweiterten Knoten).
Beispiele:
```
(MATCHP @(MATCHT GEGEN ALLES) @(MATCHT GEGEN T)) = T
(MATCHP @(A B USW USW USW) @(A B.T)) = T
```

4.4. Indem man ein Atom in eine Liste mit dem terminalen Zeichen OPTIONAL als Postfix einbettet, beschreibt man die Optionalitaet des entsprechenden Atoms.
Beispiele:
```
(MATCHP @(IMMER MANCHMAL) @(IMMER (MANCHMAL OPTIONAL))) = T
(MATCHP @(IMMER) @(IMMER (MANCHMAL OPTIONAL))) = T
(MATCHP @(IMMER IMMER) @(IMMER (MANCHMAL OPTIONAL))) = NIL
(MATCHP @(A (B C)) @(A ((B C) OPTIONAL))) = T
```

4.5. Will man ein Postfix wie OPTIONAL auch auf Teile einer Liste anwenden, so muss man den entsprechenden Teil der Liste mit dem Praefix SEGMENT versehen.
Beispiele:
```
(MATCHP @(A B C D) @(A (SEGMENT (B C) OPTIONAL) D)) = T
(MATCHP @(A D) @(A (SEGMENT (B C) OPTIONAL) D)) = T
```

4.6. Wird ein Pattern P mit dem Postfix STAR versehen, so werden in der zu pruefenden Struktur so viele aufeinanderfolgende Elemente wie moeglich mit P gematcht. Man beachte, dass erst OPTIONAL und STAR zusammen als Postfixe die Bedeutung des Kleene-Sterns haben.
Beispiele:
```
(MATCHP @(JA JA JA SO IST DAS) @((JA STAR) SO IST DAS)) = T
(MATCHP @(JA SO IST DAS) @((JA STAR) SO IST DAS)) = T
(MATCHP @(SO IST DAS) @((JA OPTIONAL STAR) SO IST DAS)) = T
(MATCHP @(ER LAEUFT UND LAEUFT UND LAEUFT !)
        @(ER LAEUFT (SEGMENT (UND LAEUFT)
                OPTIONAL STAR) !))      =     T
```

4.7. Neben dem Kleene-Stern steht dem Benutzer mit dem terminalen Zeichen OR die Vereinigung als weitere rationale Operation zur Verfuegung. Die

Praefigierung eines Atoms A mit dem terminalen Zeichen NOT bewirkt, dass alle Atome, die nicht mit A identisch sind, gematcht werden. Auf mit OR und NOT gebildete Ausdruecke kann der boolesche Operator AND angewandt werden. S-Expressions, die AND, OR und NOT in der genannten Bedeutung enthalten, werden stets mit dem Praefix PATTERN versehen in einer Liste zusammengefaßt.
Tritt innerhalb einer mit dem terminalen Zeichen PATTERN beginnenden Liste das Symbol VAR auf, so muss der auf VAR folgende Ausdruck eine Praedikatsfunktion sein.
Beispiele:

```
(MATCHP @(A B C) @(A (PATTERN (OR A B)) C)) = T
(MATCHP @(A A C) @(A (PATTERN (OR A B)) C)) = T
(MATCHP @(A B C) @(A (PATTERN (NOT B)) C)) = NIL
(MATCHP @(A B C) @(A (PATTERN (AND (NOT C) (NOT A))) C)) = T
(MATCHP @(A B C) @(A (PATTERN (AND (NOT B) (NOT A))) C)) = NIL
(MATCHP @(A B) @((PATTERN (AND (VAR (MATCHP @A @A)) A)) B)) = T
```

4.8. Um zu beschreiben, dass innerhalb der zu pruefenden Struktur n Knoten identisch sein muessen, werden im Gesamtpattern n Listen gebildet, in denen sich an das entsprechende Pattern das terminale Zeichen LABEL und n-mal das gleiche Atom als Index anschließt.
Es werden also Konstruktionen der Form
(<Pattern> LABEL <Atom>) benutzt.
Beispiele:

```
(MATCHP @(A B A) @((T LABEL BEGRENZER) (T OPTIONAL STAR)
(T LABEL BEGRENZER))) = T
(MATCHP @(Z A E I O U Z) @((T LABEL BEGRENZER) (T OPTIONAL STAR)
                        (T LABEL BEGRENZER)))= T
(MATCHP @(A B C C B A) @((T LABEL 1)(T LABEL 2)
        (T LABEL 3)(T LABEL 3)(T LABEL 2)(T LABEL 1))) = T
```

4.9. Damit auch "semantische" Kriterien in das Matching einbezogen werden koennen, wird eine syntaktische Konstruktion zur Verfuegung gestellt, mit der man das Matching vom Wert einer beliebigen Funktion abhaengig machen kann. Das terminale Zeichen FUNCTION enthaltende Ausdruecke der Form (<Atomares Pattern> FUNCTION <Funktionsname>) und (<Atomares Pattern> FUNCTION <Lambda-Expression>) bedeuten, dass das atomare Pattern P genau dann matcht, wenn die mit P als Aktualparameter aufgerufene Funktion zu T ausgewertet wird.
Beispiele:

```
(MATCHP @(A 2) @(A (T FUNCTION NUMBERP))) = T
(MATCHP @(A B) @((A FUNCTION ATOM) B)) = T
(MATCHP @(W WAHLSTER) @(T (T FUNCTION (LAMBDA (X)
            (EQ (CAR (EXPLODE X)) @W))))) = T
(MATCHP @(T WITTIG) @(T (T FUNCTION (LAMBDA (X)
                 (EQ (CAR (EXPLODE X))  @W))))) = T
```

5.1. Das dritte Argument der Funktion TRANSFORM, d.h. die nach erfolgreichem Matching durchzufuehrende Strukturveraenderung besteht im einfachsten Fall aus einer Konstante.
Beispiel:

```
(TRANSFORM @(A B) @((T T) (AKZEPTIERT))) = (AKZEPTIERT)
```

5.2. Auf die innerhalb eines Patterns durch ein mit LABEL praefigiertes Atom A markierten Teile einer zu transformierenden Struktur kann in der Beschreibung der Strukturveraenderung mit A referiert werden.

Beispiele:
```
(TRANSFORM @(A B) @(((T LABEL 1)(T LABEL 2)) (2 1))) = (B A)
(TRANSFORM @(S (NP (N COMPUTER)) (VP (V KOENNEN)
        (VP (V (RECHNEN))))
                @((S ((NP.T) LABEL 1)
                 (VP (V KOENNEN) T LABEL 2))
                 (S (VP (V KOENNEN)(NP (S 1 2))))))
= (S(VP(V KOENNEN) (NP(S(NP(N COMPUTER)) (VP(V RECHNEN))))))
```

5.3. Zum Aufbau von Beschreibungen komplexer Strukturveraenderungen dienen die terminalen Zeichen VAR und SEGMENT. Mit VAR praefigierte Ausdruecke werden im laufenden LISP-Kontext ausgewertet. Man beachte, dass die Substitution stets vor der Evaluation erfolgt, falls in einem mit VAR praefigierten Ausdruck auf Teile der zu veraendernden Struktur referiert wird. Mit dem terminalen Zeichen SEGMENT beginnende Ausdruecke bedeuten, dass das Ergebnis der durch diese S-Expression beschriebenen Strukturveraenderung in das Gesamtergebnis als Teilliste eingebettet wird.
Beispiele:
```
(TRANSFORM @(EINS ZWEI DREI) @(((T LABEL 1)(T LABEL 2)(T LABEL 3))
(1 (VAR (LIST 3 2))))) = (EINS (DREI ZWEI))
(TRANSFORM @(EINS ZWEI DREI) @(((T LABEL A)(T LABEL B)(DREI LABEL C))
(A (SEGMENT (VAR (CDR (LIST B C @VIER)))))) = (EINS DREI VIER)
```

6. Der Matching-Prozess laeuft manchmal in Sackgassen, aus denen er mit Hilfe von backtracking-Mechanismen wieder herauskommt. So laeuft der Prozess bei einem einfachen Ausdruck wie (MATCHP @(A B C D E F Z) @(A (T OPTIONAL STAR) Z)) zunaechst in eine Sackgasse, da T auch Z matcht.
Sobald erkannt wird, dass Z im Pattern noch nicht abgearbeitet ist, wird ein backtracking-Prozess ausgeloest. Um das Pattern Matching effektiver zu machen, sollte man die Bedingungen im Pattern geeignet verschaerfen. Falls im obigen Beispiel durch das Pattern mit A beginnende und mit Z endende Zeichenfolgen beliebiger Laenge erkannt werden sollen, fuehrt eine Verschaerfung des Patterns in Form von (A (PATTERN (NOT Z) OPTIONAL STAR) Z) zur Sackgassenfreiheit.
Mit (TRACE MATCHP MATCHP1 ANDLL) kann der Matching-Prozess im TRACE-Modus verfolgt werden.

7. #PATTERN wurde bei der Konstruktion eines natuerlichsprachlichen AI-Systems in dem Projekt HAM-RPM erfolgreich fuer unterschiedliche Aufgaben eingesetzt [6].

8. #PATTERN kann nur manuell als gesamtes Package geladen werden.