FACULTY OF HUMANITIES
UNIVERSITY OF COPENHAGEN

Intricate Natural Language Processing made easier with Symbolic Computation software

Pattern Matching lessons from Bracmat

Bart Jongejan, bartj@hum.ku.dk

Linguistic and algebraic expressions can be analysed with similar pattern matching (PM) methods, suggesting a trove of useful methods for Natural Language Processing (NLP).

For example, detection of the rhetoric device of parallelism is beyond the expressive power of pattern matching facilities of tools and programming languages commonly used in NLP. This poster shows how we can use a programming language for symbolic computation, Bracmat, for this task.

As explained on the right side, to detect parallelism using PM we need:

① Tree pattern matching
② Embedded expression evaluation
③ Associative pattern matching

Bracmat in a nutshell

Expressions (data, patterns and other code) are binary trees. Terminals are strings. Nodes can be prefixed. There are 15 differently behaving binary operators and 13 different prefixes.

Legend

Dotted pair	`A . B`
List concatenation	`A   B`
Specify order of operations	`(A . B) C`
Evaluate *A* and then *B*	`A & B`
Evaluate *A* or else *B*	`A | B`
Fail (forces backtracking)	`~`
Pattern matching	`subject : pattern`
Wildcard pattern, Kleene star	`?`
Bind matched value to symbol	`?symbol`
Retrieve value bound to symbol	`!symbol`
Position indicator less than *N*	`[<N`
Evaluate embedded expr. if match	`(pattern & expression)`
Insert binding without evaluation	`() ' (… () $ symbol …)`
Bind unevaluated expression	`symbol = expression`
Call a (pattern) function	`fnc $ argument`
Function definition	`fnc = local-vars . body`
Function argument	`!arg`
Exta argument in pattern function	`!sjt`

Term rewriting systems support ① and sometimes ③. XQuery, XSLT, Trafola (Heckmann, 1988), Elan (Borovansky et al., 1997), Tom (Moreau et al., 2003), and Tregex+Tsurgeon (Levy and Andrew, 2006), Maude (Clavel et al., 1998), Stratego (Visser, 2001)

Functional languages support ① and ②.

Perl supports ② (embedded code execution) and ③.

Egison (Egi, 2014) supports ①,②,③ + commutative PM.

Bracmat (github.com/BartJongejan/Bracmat) supports ①,②,③ and automatically normalizes expressions containing commutative operators (+ and *). This partly compensates for the lack of commutative PM.

Peter Borovansky, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, and Marian Vittek. 1997. Elan: A logical framework based on computational systems. Elsevier.

M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. Quesada. 1998. Maude as a metalanguage. In *2nd International Workshop on Rewriting Logic and its Applications (WRLA'98)*, volume 15 of *Electronic Notes in Theoretical Computer Science*. Elsevier.

Satoshi Egi. 2014. Non-linear pattern-matching against unfree data types with lexical scoping. *CoRR*, abs/1407.0729.

Reinhold Heckmann. 1988. A functional language for the specification of complex tree transformations. In H. Ganzinger, editor, ESOP '88, volume 300 of *Lecture Notes in Computer Science*, pages 175–190. Springer Berlin Heidelberg.

Roger Levy and Galen Andrew. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *5th International Conference on Language Resources and Evaluation*.

Pierre-Etienne Moreau, Christophe Ringeissen, and Marian Vittek. 2003. A pattern matching compiler for multiple target languages. In *12th Conference on Compiler Construction, Warsaw (Poland), volume 2622 of LNCS*, pages 61–76. Springer.

James R. Slagle. 1974. Automated theorem-proving for theories with simplifiers commutativity, and associativity. *J. ACM*, 21(4):622–642, October.

Eelco Visser. 2001. Stratego: A language for program transformation based on rewriting strategies. System description of Stratego 0.5. In A. Middeldorp, editor, *Rewriting Techniques and Applications (RTA'01)*, volume 2051 of *Lecture Notes in Computer Science*, pages 357–361. Springer-Verlag, May.

Experiment: Detect parallelism

Find every constituent in the Penn treebank that contains a repeating group of labels. We do not specify which labels a repeating group must contain, but require that the group occurs twice and that at most two labels in the constituent are not included in the repeating group.

① Tree pattern matching

The subject matter in our experiment is a tree structure, and so are patterns that match the subject matter.

Bracmat patterns (coloured blue in all figures) always match the whole subject, but not necessarily in full detail, see Fig. 1. The unspecified parts are matched by wildcards `?` or variables `?symbol`.

An example of Penn Tree data represented in Bracmat is shown in Fig. 2, together with a matching tree pattern.

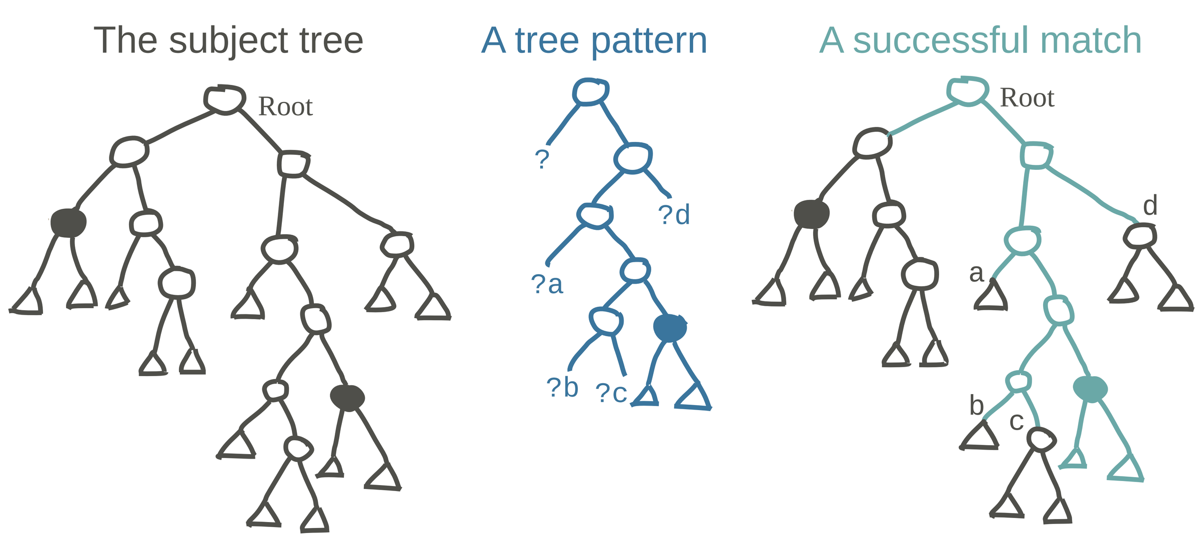

Fig.1. Tree pattern matching in Bracmat, schematic.

```
( S
.   (CC.So)
    (NP-SBJ.PRP.he)
    ( VP
    .   ( VP
        . (VBD.sought) (NP.PRP.her) (PRT.RP.out)
        )
        (",".",")
        (CC.and)
        ( VP
        . (VBD.spoke) (PP.(TO.to) (NP.PRP.her))
        )
        (",".",")
        (CC.and)
        ( VP
        .   (VBD.thought)
            ( PP
            .   (IN.of)
                ( NP
                .   (NP.("PRP$".his) (NN.hand))
                    ( PP
                    .   (IN.in)
                        (NP.("PRP$".her) (NN.hair))
                    )
                )
            )
        )
    )
    (".".".")
)
: (S.? (?.?C.he) (VP.?vp) (?punct."."))
```

Fig. 2. Sentence from Penn Treebank and a matching pattern.

② Embedded expression evaluation

When searching for repeating groups of category labels we do not a priori know what exactly to look for. All we can do is create a general 'group of labels' pattern that requires that there must be at least two labels.

When a candidate group `grp` is found - a situation that can occur many times when applying the general pattern to a syntactic analysis of a sentence – pattern matching must temporarily yield to the creation of a more specific subpattern that thereafter is used to search for the second occurrence of the candidate group of labels in the remaining part of the subject, see Fig. 3.

An embedded expression can act as a guard, forcing the pattern matching operation to backtrack or fail if some condition is not met. It can also be used for its side effects, such as accumulation of results. Fig. 4 illustrates both.

Pattern matching and expression evaluation can be nested many levels deep.

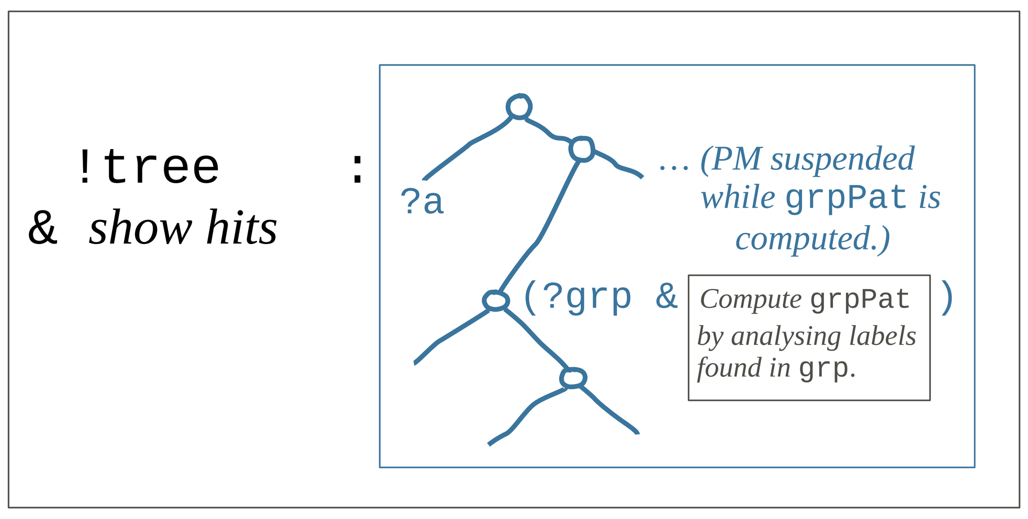

Fig. 3. Embedding expression evaluation in pattern matching *v.v.*

```
{?}   :?Ls
    & (   (DT.a) (JJ.one-time) (NN.actor)
        : ? (?L&!Ls !L:?Ls&~.?) ?
      | !Ls
      )

{!} DT JJ NN
```

Fig. 4. Collect found labels in variable `Ls` while scanning the subject phrase.

③ Associative pattern matching

Candidate groups of labels can start in several positions and end an unknown number of labels later. The pattern matcher therefore must slice the currently investigated constituent in different ways (indicated by the orange lines and arrows in Fig. 5) until a partition is found with two slices with the same labels, or until all partitions have been tried. This process is associative PM (Slagle, 1974). Associative PM is well known from regular expressions applied to strings of characters, but here we have a list of trees.

Lists can be represented as binary tree structures. They can occur as subtrees in otherwise hierarchically structured trees, as is the case in a treebank.

Associative PM requires an associative operator. For strings and lists this operation is concatenation, with neutral element `""` or `()`. Other associative operators are addition and multiplication, with neutral elements `0` and `1`.

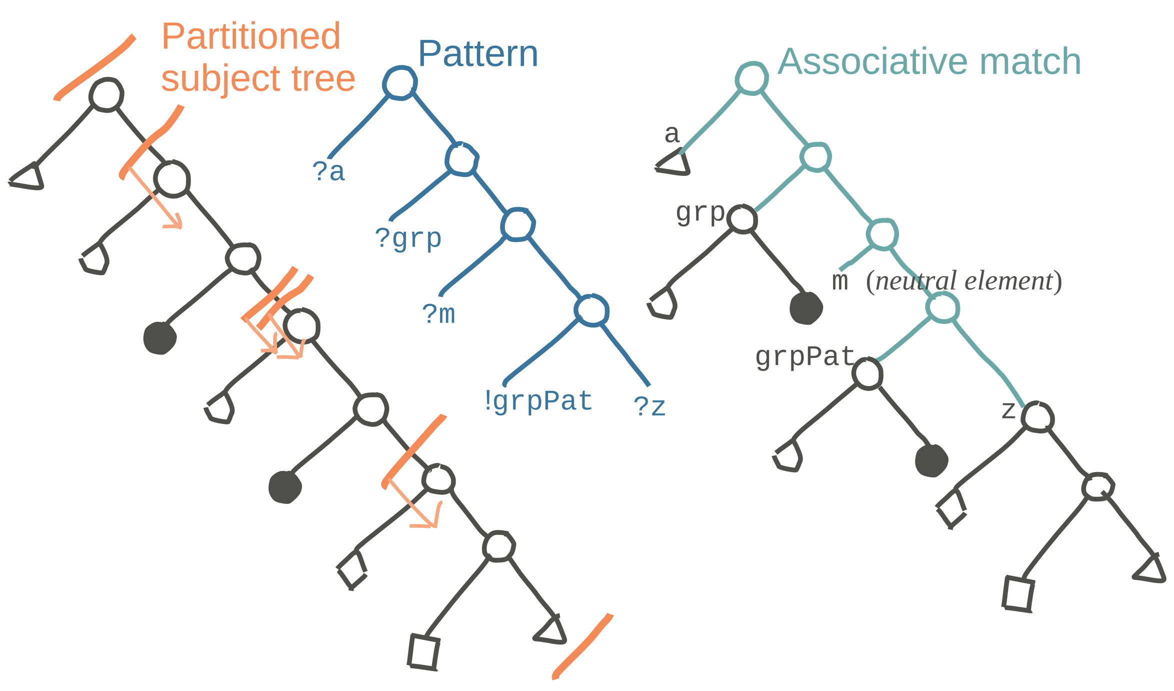

Fig. 5. Partition that isolates two groups with identical labels.

The complete program for detecting and listing the constituents with repeating groups is shown in Fig. 6.

The treebank is matched against `pattern`, which inspects all sentences one by one, driven by forced backtracking.

```
  ( has
  = .   '(()$sjt) : (!arg|(=?.? ((?.?):has$!arg) ?))
      & ?
  )
& ( pattern
  =   ?
      ( (?.?)
      :   has
        $ (
          =   ?Tag
            .     ?a
                  ( (?.?) (?.?) ?:?grp
                  & :?grpPat
                  &   whl
                    ' ( !grp:?grp (?Lbl.?)
                      & (!Lbl.?) !grpPat:?grpPat
                      )
                  )
                  ?m
                  !grpPat
                  (?z&!a !m !z:? [<3)
              : ( ?Phrase
                & (!grpPat.!Tag.!Phrase) !Acc:?Acc
                & ~
                )
          )
      )
      ?
  )
& :?Acc
& get$"input/pennIII.bra"
& (!NTrees:!pattern|lst$(Acc,"report.txt",NEW));
```

Fig. 6. Bracmat program that detects repeating groups of labels in the Penn Treebank.

Result

12 000 hits in 250 000 sentences, e.g., VP constituent that contains a repeating group with pattern `grpPat = (VP.?) (",".?) (CC.?)` :

a — grp — m — grpPat — z

So he [|| [sought her out]$_{VP}$ [,]$_{,}$ [and]$_{CC}$ || [spoke to her]$_{VP}$ [,]$_{,}$ [and]$_{CC}$ | [thought of his hand in her hair]$_{VP}$ |]$_{VP}$.

Niels Klim and tree-people of Potu (1845 English edition of Niels Klim's Journey Under the Ground)