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

**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.

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 label-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.

**Embedded expression evaluation**

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.

**Associative pattern matching**

Candidate groups of labels can start in several positions and end an unknown number of labels later. The pattern matcher therefore has to retrieve 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 (Stalinger, 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 $.

Other associative operators are addition and multiplication, with neutral elements 0 and 1.

**Conclusion**

An embedding 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.

**Pattern matching**

Candidate groups of labels can start in several positions and end an unknown number of labels later. The pattern matcher therefore has to retrieve 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 (Stalinger, 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 sub trees 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 $.

Other associative operators are addition and multiplication, with neutral elements 0 and 1.

**Conclusion**

A tree pattern matching

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.

**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.

**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.

**Term rewriting systems support** 1 and 2

1. Pattern matching
2. Embedded expression evaluation
3. Associative pattern matching

**Functional languages support** 1 and 2

1. Pattern matching
2. Embedded expression evaluation
3. Associative pattern matching

**Bracmat** (which includes Bracmat Jr. Bracmat Jr.) supports 1, 2 & 3 and automatically normalizes expressions containing associative operators (+ and *). This partly compensates for the lack of commutative PM.