MonaSearch – Querying Treebanks with Monadic Second Order Logic

Hendrik Maryns

Sonderforschungsbereich 441
Universität Tübingen
hendrik@sfs.uni-tuebingen.de

2008-09-19

Eberhard Karls
Universität
Tübingen

Outline

Outline

Outline

Goal

- Offer linguists the possibility to query big tree banks.
- Ideally: high expressive power, fast and easy to use.
- Use of a logical formalism: clearly defined, easy to understand (but...).
- Monadic Second Order logic (MSO) chosen, often considered as the expressive power needed in linguistics.
- Implemented using tree automata.

Übersicht

Treebanks

- Useful, big.
- No tool yet offers possibility to query automated treebanks with millions of sentences.

Considerations about querying

- We want as few results for a query as possible: only the results that are relevant should be returned.
- This asks for a formalism that allows one to describe exactly what is needed. A high expressive power is wanted.
- The relevant factor is the data complexity: the size of the treebanks justifies precompilation and a bigger complexity in processing the query.

Übersicht

Monadic Second Order logic

- Extension of first order logic.
- Addition of second-order variables, representing node sets.
- Quantification allowed over variables of both first and second order.

Example:

$\exists X(\neg\exists x(x \in X))$ 'There exists an empty set.'

Properties:

- Higher expressive power compared to FOL
- Transitivity of binary relations can be expressed
- Clear semantics: the usual model-theoretical semantics

Tree Automata

- Thatcher and Wright and Doner [3, 1] describe how an MSO formula can be translated into a tree automaton.
- Unfortunately, the process is EXPTIME complete in the length of the formula.
- OTOH, the evaluation of a tree automaton on a tree is linear in the size of a tree.

Noting that

1. the data complexity is by far the most important factor and
2. no overly complex queries are expected from linguists

it is promising to use this as an algorithm for querying treebanks.

Implementation

A first try to implement the automaton operations ourselves has failed due to memory problems.
There is, however, a program that implements the translation from MSO to tree automata: MONA [2]. MONA was developed for hardware verification and is therefore not immediately usable for our purposes. It verifies formulas, computing the automaton in the background. Fortunately, they offer a small library providing functions to evaluate an automaton on a tree.
Big thanks go to Michael Jakl from Vienna, who pointed this possibility out to us. (The user manual is *very* concise on this point.)

Übersicht

Querying

The basic steps in querying are the following:

Querying

The basic steps in querying are the following:

1. The user enters a formula, expressing a property of the wanted tree.

Querying

The basic steps in querying are the following:

1. The user enters a formula, expressing a property of the wanted tree.
2. The formula is translated into a tree automaton by MONA.

Querying

The basic steps in querying are the following:

1. The user enters a formula, expressing a property of the wanted tree.
2. The formula is translated into a tree automaton by MONA.
3. Every tree is checked with the automaton.

Querying

The basic steps in querying are the following:

1. The user enters a formula, expressing a property of the wanted tree.
2. The formula is translated into a tree automaton by MONA.
3. Every tree is checked with the automaton.
4. The trees fulfilling the formula are recognized by the automaton and delivered to the user.

Problems

- Fundamental:
  - MONA can only handle binary trees; input trees can have arbitrary branching, are often not real trees
  - MONA expects nodes to be labeled with bit vectors
  - MONA defines its own variant of tree automata, Guided Tree Automata.
- Technical:
  - MONA must be called externally
  - The MONA library is written in badly documented C, the GUI and preprocessor in Java.

Übersicht

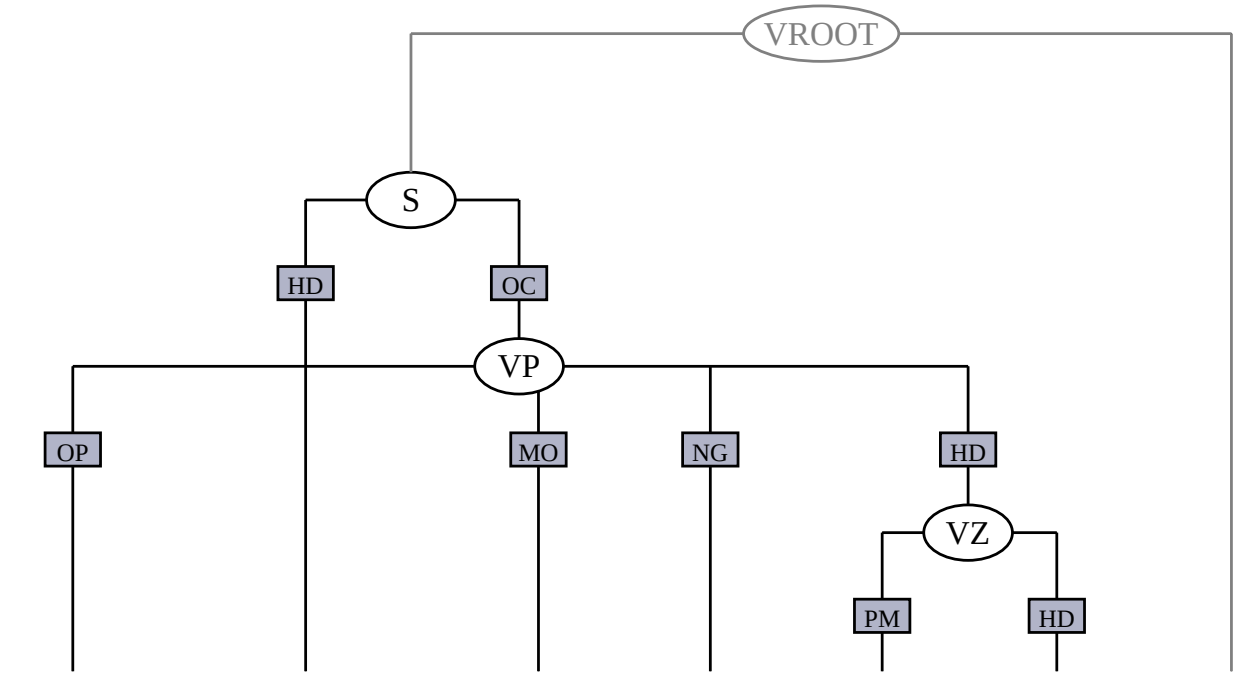
Trouble with treebanks

The name ‘treebank’ really is a euphemism:

1. Some sentences contain multiple non-connected subtrees (often for punctuation, but also for subclauses);
2. not only are they disconnected, they can be embedded in each other;
3. crossing edges occur;
4. due to secondary edges, the ‘trees’ are actually DAGs.

Problems 2 and 3 are caused by the additional order on the leaves of the trees.

Example tree with crossing edges from the TIGER treebank



Binarization

A classical trick is used to transform arbitrary branching trees into binary trees: "first daughter, next sister". A node x in the original tree is transformed into node x' in the binary tree as follows:

- If x has children, call its first daughter y, then y' becomes the left child of x'.
- If x has any siblings, call its next sister z, then z' becomes the right child of x'.
- Nodes keep their labels; edge labels are stored at the node below. (Each node can have multiple labels.)

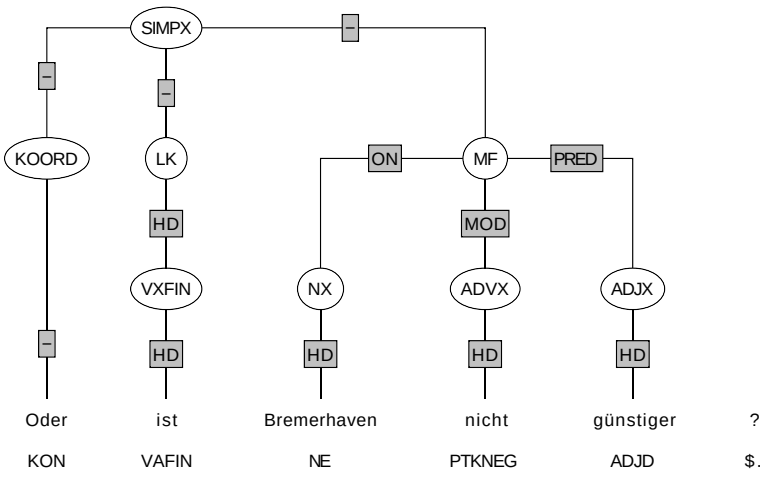
Further preprocessing

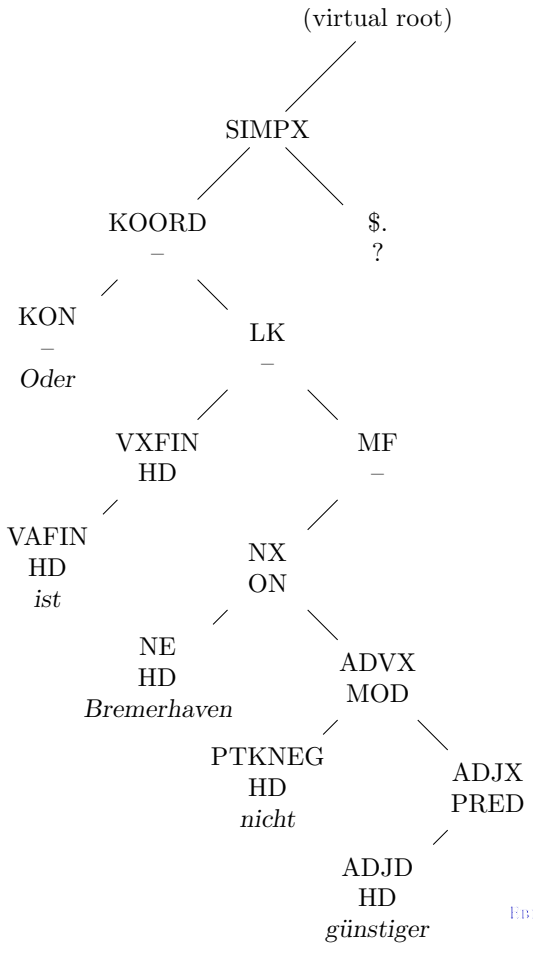
Some other problems are handled in preprocessing:

- A virtual root node is introduced, at which all disconnected parts are attached.
- Crossing edges are resolved in that only the order of the daughters as seen by their mother is considered. I.e. the imposed order on the leaves is given up.
- Secondary edges are ignored.

Example

The tree on the left is transformed into the tree on the right:





Formula transformation

The user formulates his query ϕ on the original tree b. To ensure correctness, we have to transform it such that the new formula ϕ' is valid in the binary tree b' iff ϕ is valid in b.
At the same time, the formula entered by the user must be braught in a format MONA can handle. These two steps are combined, thus taking away the need for the user to learn the idiosyncratic MONA language.
Most connectors can be translated directly into their MONA counterpart, but relations concerning the structure of the tree must be translated into more complex MONA formulas.

Example

The formula $x \triangleleft y$, 'x is the mother of y', is transformed to

```
ex1 z: x.0 = z & right_branch(z,y)
```

where `right_branch(a,b)` is a formula expressing b is on the right branch starting at a:

```
pred right_branch (var1 x,y) =
  all2 X: (x in X &
    (all1 z, w: (z in X & w = z.1) => w in X)
  ) => y in X;
```


Übersicht

Labels in the treebank

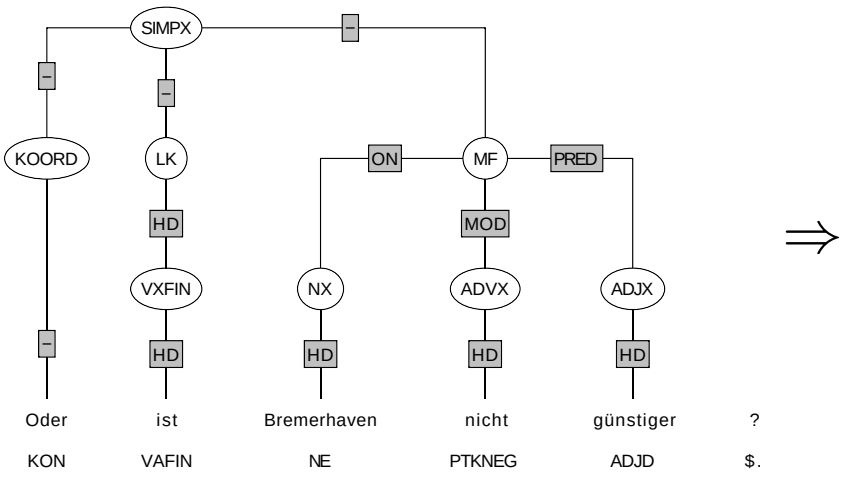
The labels in the treebank represent grammatical functions of the subtrees. In some formats, edge labels are present and leaves can have multiple labels (word, morphology, POS, ...).
The MSO translation offers us a simple way to handle this: labels are handled as predicates. A label is thus identified with the set of all nodes bearing the label. This way, it is no problem for a node to have multiple labels.
To enable MONA to handle the labels, they are coded as bit vectors: the labels occurring in the formula receive an index number in order of appearance; then, a node in the binary tree has a 1 at the index of X if it bears the label X, otherwise it gets a 0.
MONA treats labels as free second-order variables. They are coded in the automaton with the same bit vectors.

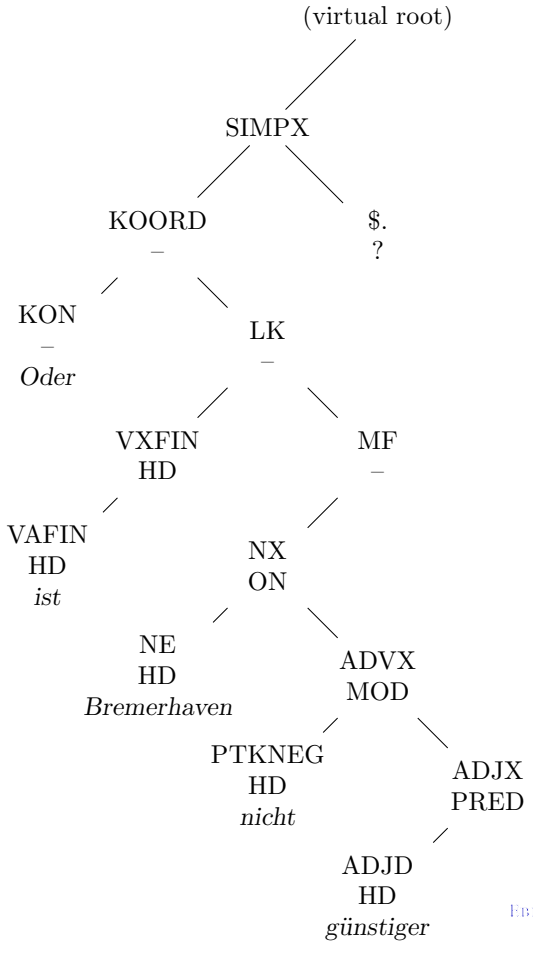
Steps taken

Note that the encoding of the labels depends on the input formula. This implies it has to be recomputed for each formula. The binarization, on the other hand, is independent of the formula and must therefore be done only once.
At registering a new treebank for querying, a preprocessing step launches which does the first transformation, but still keeps the original labels.
The computation of the bit vectors is done on the fly for each submitted query, before the trees are handed over to the automaton.

Example

The tree on the left is transformed into the tree on the right:





Übersicht

Difference to regular tree automata

Guided Tree Automata are tree automata with a guide. The guide expresses knowledge about the problem at hand, about the model. This can help to restrict the size of the automata computed.
In our case, this is of not much help, since we know nothing beforehand about the model: all trees are different.

A lucky coincidence

It is possible to tell MONA to generate a dummy guide. As a consequence, the root node and its right subtree are treated as dummies; the real tree to be queried is expected as the left subtree of the root.
But remember that at preprocessing, we introduced a virtual root node to connect disconnected subparts. After binarization, this virtual root node will be the root node of the binary tree and the root of the real tree will effectively be its left daughter.

Übersicht

Calling MONA

The formula from the user is converted into a format MONA understands and written in a file. Then MONA is called with a system call on the file and the automaton it outputs is gathered in another text file.

Disadvantage: MONA must be installed on the client computer (and be located on the path).

Running the automaton

The MONA library, written in C, is called with JNI. SWIG is used to simplify this process, it creates wrapper files for the necessary C structures.
The automaton is read from the file with a library function. The bit vectors of the binary trees are computed on the Java side and a binary tree in C is constructed with the wrappers. This C tree is checked by the automaton.

Disadvantage: JNI requires platform dependent libraries, the platform independence from Java is lost.
However, by using Java Web Start, the user need not be bothered by this.

Demo

`http://tcl.sfs.uni-tuebingen.de/MonaSearch`

Outlook

Extensions and enhancements in progress:

- Offer a visualization component with browsable treebanks.
- Handle more complex structures in the treebank: secondary edges etc.
- GUIs can always be enhanced.

FINIS

Thank you for listening.

References


John E. Doner.
Tree acceptors and some of their applications.
Journal of Computer and System Science, 4:406–451, 1970.


Nils Klarlund and Anders Møller.
MONA Version 1.4 User Manual.
BRICS, Department of Computer Science, University of Aarhus, January 2001.
Notes Series NS-01-1. Revision of BRICS NS-98-3.


James W. Thatcher and Jesse B. Wright.
Generalized finite automata theory with an application to a decision problem of second-order logic.
Mathematical Systems Theory, 2(1):57–81, 1968.