MonaSearch – Querying Treebanks with Monadic Second Order Logic

Who? Hendrik Maryns

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

When? TLT'09

Eberhard Karls
Universität
Tübingen

Overview

MSO

Overview

MSO

Tree automata

Overview

MSO

Tree automata

Problems?

Overview

MSO

Tree automata

Problems?

Transitive Closure

Overview

MSO

Tree automata

Problems?

Transitive Closure

How it works

Overview

MSO

Tree automata

Problems?

Transitive Closure

How it works

Conclusion and Outlook

Motivation

Large treebanks, databases containing linguistic annotations of real-world example sentences in the form of trees, have been compiled over the last years.
Hand-annotated treebanks contain tens of thousands of trees, but there are automatically-tagged corpora with millions of sentences available.
It is obvious one cannot look for certain phenomena in these treebanks by hand. Tools are needed which facilitate this process.
A key observation here is that it is important to keep result sets small: one wants to see exactly those sentences containing the phenomenon one is looking for. This asks for a powerful query language.

Implementation Goals

- Provide a powerful query language
- Clearly defined semantics
- Independent of particular annotation or form of treebank
- Easy to use

Overview

MSO

Negation

One annoyance with the currently available systems is the limited possibility to use negation. Imagine one looks for a verb phrase without a dative object. This query could look as follows:

$$\exists x(\mathrm{VP}(x) \wedge \neg\exists z(x \lhd z \wedge \mathrm{DA}(z)))$$

Note the negation which is not on the atomic level, but negates a whole subformula. Because of this, this query is not expressible in the majority of existing query tools.

Quantifiers

Another logical construction most query tools have problems with is quantification. Most of the time, nodes that are declared are implicitly quantified existentially.
Universal quantification is not available.
This is closely related to the availability of negation, since $\forall = \neg\exists\neg$. Thus, the previous query would be an example of this too.
Some tools offer universal quantification, but without negation.

Solution

MonaSearch uses a well-known logic formalism as query language: Monadic Second-Order Logic (MSO).

- Clearly defined, model-theoretic semantics
- High expressive power
- Any tree structure can be queried.

The idea and the way to realize this stem from Kepser [4], which forms the theoretical base for MonaSearch.

The power of MSO

- MSO is an extension of first-order predicate logic by second-order variables. These are interpreted to range over (finite) sets.
- As such, full negation is available.
- It is even possible to express the transitive closure of a relation. This allows one to look for e.g. recursive patterns.

With MSO as query language, negation is handled effortlessly.

Tree automata

Overview

Formulation Problem

fsq ([3]) offers the full power of first order logic, and is thus one of the few query tools which can handle negation as above. It suffers from another problem, however: the query time is highly dependent of the way in which the query is posed. Different formulations of the same query, which are logically equivalent, can have vastly different query times.

Tree Automata

The implementation of MonaSearch is based on a famous result from the 60s [5], that states that MSO can be translated into tree automata. Tree automata, once computed, have a linear evaluation time on a tree. This approach was exploited by the tool MONA [1], designed for hardware verification. MONA can be used to compute the tree automaton to a given formula. MonaSearch wraps MONA to offer an interface suitable for linguistic use.

Normal Form

The tree automata function as a logical form for the query and therefore MonaSearch does not suffer from the formulation problem. The exact formulation is irrelevant: it will always compute the same tree automaton.
The user, however, does not need to know anything about tree automata, the process is entirely transparent.

Overview

Problems?

Decidability

MSO in general is undecidable. The equivalence with tree automata provides decidability on trees.
This has the nice side-effect that if a query returns no results, it is guaranteed that there are none.
By exploiting several tricks, decidability can be established on finite structures. A general procedure can be found in [2].
This makes it possible to tackle the structures in tree banks which go beyond treeness: secondary edges, crossing edges and disconnected substructures. The general method has not been implemented, but some simpler preprocessing takes care of disconnected parts and crossing edges.

Complexity

The translation of MSO into tree automata has a very high complexity. For each quantifier change (that is, an existential quantifier followed by a universal quantifier, or the converse), an exponential step is made.
In practice, this turns out not to be a problem. Of all the queries we have tried out, none took longer than a few milliseconds for the computation of the automaton. Generally, for linguistic purposes, no queries are to be expected that would reach the limit of computability.

Overview

Transitive Closure

Second Order

MSO transcends first-order logic by allowing second-order variables. This allows one to express the recursive closure over a relation expressed as a formula. Imagine looking for PPs that are arbitrarily deeply nested. The basic formula would be:

$$R(x, y) \equiv x \lhd y \wedge \mathrm{PP}(x) \wedge \mathrm{PP}(y)$$

We can then insert this formula into the template which calculates the transitive closure of a formula:

$$\forall X(\forall z(R(x, z) \rightarrow z \in X) \wedge (\forall z, w(z \in X \wedge R(z, w) \rightarrow w \in X)) \rightarrow y \in X).$$

Overview

How it works

Step by step

- a tree bank is selected
- the tree bank is preprocessed into binary trees; this is done only once, the result is stored in a file
- the user composes a query
- the query is transformed to be evaluated on the binary trees and converted into something MONA understands
- MONA is invoked on the query to compute the automaton
- the automaton is run on every binary tree
- the id of each tree recognized is returned, the linguistic tree with that id satisfies the original query
- the results are presented to the user

More details in the article.

Overview

Conclusion and Outlook

Conclusion

- provided a powerful query tool
- does not suffer from the formulation problem
- efficiency at least as good as existing tools
- linear lookup: querying of automatically annotated corpora seems feasible

MonaSearch as a Preprocessor?

The automaton approach brings with it the weakness that only whole trees can be returned as results. But MonaSearch can be used as some sort of preprocessor, the result of which is then piped into (slower, less powerful) tools which look in MonaSearch's results to present the user with more detailed information.
Due to the high expressive power of MSO, it should be no problem to wrap a query in the lesser expressive language, the second-level tool uses.

Enhancements

There is still a lot of work which could be done. Apart from the user interface, which can of course always be enhanced, some major enhancements are thought of:

- A visualization component with which one can choose a sentence from the results to see the corresponding tree.
- An indexing component which does pre-indexing on node labels occurring in the formula.
- Handling of disconnected trees and secondary edges following the method in the article by Flum, Frick and Grohe. Involves additional work in the several translation steps, but some parts can probably be handled easier.

A very uncertain future

The project at which MonaSearch was developed finished end of 2008. I will occasionally still work on it, making minor improvements and fixing bugs. However, no major work will be done.
If anybody is interested in continuing this project or integrating it into a bigger program, let me know!

FINIS

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

References I

Jacob Elgaard, Nils Klarlund, and Anders Møller.
Mona 1.x: new techniques for ws1s and ws2s.
In *Computer Aided Verification, CAV '98, Proceedings*, volume 1427 of *LNCS*. Springer Verlag, 1998.

Jörg Flum, Markus Frick, and Martin Grohe.
Query evaluation via tree-decompositions.
Journal of the ACM, 49:716–752, nov 2002.

Stephan Kepser.
Finite structure query: A tool for querying syntactically annotated corpora.
In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, volume 1, page 179–186, Budapest, 2003.

References II

Stephan Kepser.
Querying linguistic treebanks with monadic second-order logic in linear time.
Journal of Logic, Language, and Information, 13:457–470, 2004.

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.