

## Chapter 1

---

# Querying Linguistic Treebanks with Monadic Second-Order Logic in Linear Time

STEPHAN KEPSEK  
SFB 441, UNIVERSITY OF TÜBINGEN, GERMANY  
KEPSEK@SFS.UNI-TUEBINGEN.DE

## 1.1 Introduction

In recent years large amounts of electronic texts have become available providing a new base for empirical studies in linguistics and offering a chance to linguists to compare their theories with large amounts of utterances from “the real world”. While tagging with morphosyntactic categories has become a standard for almost all corpora, more and more of them are nowadays annotated with refined syntactic information. Examples are the Penn Treebank (Marcus et al., 1993) for American English annotated at the University of Pennsylvania, the French treebank (Abeillé and Clément, 1999) developed in Paris, the NEGRA Corpus (Brants et al., 1999) for German annotated at the University of Saarbrücken, the Tübingen Treebanks (Hinrichs et al., 2000) for Japanese, German and English from the University of Tübingen, and the German newspaper corpus TIGER (Brants et al., 2002). To make these rich syntactic annotations accessible for linguists the development of powerful query tools is an obvious need and has become an important task in computational linguistics.

Consequently, a number of query tools for syntactically annotated corpora have been developed in recent times. Amongst the most important ones are CorpusSearch (Randall, 2000), fsq (Kepser, 2003), ICECUP III (Wallis and Nelson, 2000), TGrep2 (Rohde, 2001), TIGERsearch (König and Lezius, 2000), and VIQTORYA (Kallmeyer and Steiner, 2002). All of them face a fundamental problem in the design of a query system namely the definition of the expressive power of the query language. The problem lies in balancing out user demands for a high expressive power on the one hand and complexity problems on the other that may arise when query languages become quite powerful. The difficulty of this problem grows with the fact that the so-called treebanks to be queried are very often not just collections of proper trees. Demands of linguists have introduced additional features such as crossing branches and secondary relations. In consequence, some treebanks have more or less become collections of

finite structures. Most of the above mentioned query tools (namely CorpusSearch, ICECUP III, TGreg2, and VIQTORYA) ignore this additional challenge completely, they are designed to query trees only. Still, they typically offer query languages of limited expressive power with the existential fragment of first-order logic being a kind of upper bound. A notable exception is fsq, which was particularly developed to query arbitrary finite first-order structures with full first-order logic. The disadvantage of fsq is the complexity of the implemented algorithm: Evaluation time of a query is polynomial in the size of the treebank. The size of the lead polynome is the quantifier depth of the query. Hence the evaluation of complex queries can take quite a long time.

Building on insights from theoretical informatics we show here that it is possible to query linguistic treebanks with a monadic second-order logic, powerful query language, in time linear in the size of the treebank.

It is known that the evaluation of a first-order sentence on a finite structure with a carrier of just two elements is already PSPACE-complete. To differentiate between the contribution of the logic or query language on the one hand and the contribution of the size of the finite structure on the other, Vardi (1982) introduced the notions of *query complexity* and *data complexity*. The general situation in querying linguistic treebanks is such that treebanks are large and still growing huger while most queries are relatively small. This justifies the concentration on data complexity, as we do it here.

## 1.2 The Query Language

The query language we propose is monadic second-order logic (MSO). It is the extension of first-order logic by set variables. As stated above, “trees” in a treebank may contain unconnected subparts and directed as well as undirected secondary edges. We therefore see a tree in a treebank as a finite relational structure. Technically, we follow the exposition by Arnborg, Lagergren, and Seese (1991). The signature of a tree consists of the unary predicate symbols  $V, E, D, P_1, \dots, P_p$  for some  $p \in \mathbb{N}$  and of the three binary predicates  $R_1, R_2, R_3$  with the intended meaning that

- $V$  designates the set of vertices,
- $E$  designates the set of undirected edges,
- $D$  designates the set of directed edges,
- $R_1(a, b)$  holds if and only if  $a$  is a vertex incident with the edge  $b$ ,
- $R_2(a, b)$  holds if and only if  $a$  is the source or origin of the directed edge  $b$ ,
- $R_3(a, b)$  holds if and only if  $a$  is the target or end point of the directed edge  $b$ ,
- $P_1, \dots, P_p$  are (linguistic) labels of vertices and edges.

Let  $\mathcal{V}'_1 = \{x, y, z, x_1, x_2, x_3, \dots\}$  and  $\mathcal{V}'_2 = \{X, Y, Z, X_1, X_2, X_3, \dots\}$  be two disjoint denumerable sets of individual variables (vertices or edges) and set variables. We use lower case letters for individual variables and upper case letters for set variables. The syntax of MSO contains two binary logical relations, namely = (equality) and

$\in$  (membership). Formulae are defined as follows. For all  $x, y \in V_1, X \in \mathcal{V}_2, 1 \leq j \leq p$ :  $V(x), E(x), D(x), P_j(x), R_1(x, y), R_2(x, y), R_3(x, y), x = y, x \in X$  are atomic formulae. Let  $\phi$  and  $\psi$  be formulae. Then  $\neg\phi, \phi \wedge \psi, \phi \vee \psi, \phi \rightarrow \psi, \exists x\phi, \forall x\phi, \exists X\phi, \forall X\phi$  are formulae.

A tree is a finite relational structure  $\mathcal{T} = (U, V, E, D, R_1, R_2, R_3, P_1, \dots, P_p)$  where  $U$  is a finite nonempty set of vertices and edges,  $V, E, D$  are unary predicates of vertices and (undirected and directed) edges,  $P_1, \dots, P_p$  are unary predicates, and  $R_1, R_2, R_3$  are binary predicates. We call such a structure also a graph. The size of the graph is  $|U|$ , the number of vertices and edges of the graph.

The semantics of MSO over these structures is an extension of the classical first-order logic semantics. A variable assignment  $a$  now consists of two functions  $a_1 : \mathcal{V}_1 \rightarrow U$  and  $a_2 : \mathcal{V}_2 \rightarrow \wp(U)$ . Formulae not involving set variables have the same semantics as in the first-order case. The formula  $x \in X$  is true iff  $a(x) \in a(X)$ . For set quantification,  $\exists X\phi$  is true in  $(\mathcal{T}, a)$  iff there is a  $W \subseteq U$  such that  $\phi$  is true in  $(\mathcal{T}, a[X/W])$  where  $a[X/W]$  is a variable assignment that is equal to  $a$  except that it assigns  $W$  to  $X$ . The formula  $\forall X\phi$  is true in  $(\mathcal{T}, a)$  iff for all  $W \subseteq U$  the formula  $\phi$  is true in  $(\mathcal{T}, a[X/W])$ .

Not every finite structure of the given signature is suitable to designate a tree. To ensure the intended meaning of the predicates some axioms have to be added that all structures should respect.

$$\begin{aligned} \forall x V(x) \vee E(x) \vee D(x), & \quad \forall x \neg(V(x) \wedge E(x)), \\ \forall x \neg(E(x) \wedge D(x)), & \quad \forall x \neg(V(x) \wedge D(x)), \\ \forall x, y R_1(x, y) \rightarrow (V(x) \wedge E(y)), & \quad \forall x, y R_2(x, y) \rightarrow (V(x) \wedge D(y)), \\ \forall x, y R_3(x, y) \rightarrow (V(x) \wedge D(y)). & \end{aligned}$$

The axioms state that the relations  $V, E$  and  $D$  partition the domain  $U$  and that the left hand side argument of a relation  $R_{1,2,3}$  is always a vertex while the right hand side is an undirected edge for  $R_1$  and a directed edge for  $R_{2,3}$ .

A linguistic treebank in our sense is a finite set of finite structures defined as above. Restricting ourselves to finite treebanks is justified on two grounds. Firstly, any now or in the future existing treebank is finite. Secondly, querying an infinite treebank makes no sense since a person posing a query expects an answer at least in finite time.

### 1.3 Linear Time Complexity of MSO Queries

In the general case, the data complexity of MSO queries on arbitrary classes of finite structures is PSPACE (see, e.g., Ebbinghaus and Flum (1995)). Thus there is little hope to find efficient algorithms for MSO queries on arbitrary classes of finite structures. But there is a class of structures for which a linear-time algorithm exists. As was shown independently by Arnborg, Lagergren, and Seese (1991) and by Courcelle (1990a,b, 1992), the class of graphs *with bounded treewidth* possesses such an algorithm.

The notion of *treewidth* was introduced by Robertson and Seymour (1986) as a way to measure how close to a tree a graph is. Bodlaender (1993) provided a general introduction that the interested reader is referred to.

**Definition 1** A *tree decomposition* of a graph  $(A, V, E, D, R_1, R_2, R_3, P_1, \dots, P_p)$  is a

pair  $(T,S)$ , where  $T$  is a tree and  $S$  is a family of sets indexed by the vertices of  $T$  such that

1.  $\bigcup_{X_t \in S} X_t = A$ .
2. For all  $c \in A$  such that  $E(c)$  there is a unique  $X_t \in S$  such that  $c \in X_t$ , and if  $a \in A$  satisfies  $R_1(a,c)$  then  $a \in X_t$ .
3. For all  $c \in A$  such that  $D(c)$  there is a unique  $X_t \in S$  such that  $c \in X_t$ , and if  $a \in A$  satisfies  $R_2(a,c)$  or  $R_3(a,c)$  then  $a \in X_t$ .
4. For all  $a \in A$  the subgraph of  $T$  induced by  $\{t \mid a \in X_t\}$  is connected.

The *width* of such a decomposition is  $\max_{X_t \in S} |\{a \mid a \in X_t, V(a)\}| - 1$ , i.e., the largest number of vertices in a single set of the decomposition minus 1.

A graph  $G$  is of *treewidth*  $k$  if and only if the smallest width of a tree decomposition of  $G$  is  $k$ .

There are different ways in which a graph can divert from a tree. A clique, for example, is a structure which is kind of an opposite of a tree. Hence it is simple to see that the size of the largest clique is a lower bound of the treewidth of a graph. An important property of a tree is that every pair of vertices of a tree is connected by a *unique* path. A graph in which many pairs of vertices are connected by many different independent paths is therefore also kind of an opposite of a tree. The largest number of independent paths between two vertices gives an upper bound for the treewidth.

**Proposition 1** Arnborg et al. (1991); Courcelle (1990a,b, 1992) *For every class  $K$  of graphs of universally bounded treewidth, every MSO sentence can be decided in time linear in the size of  $G$  for  $G \in K$ .*

A fortiori, every finite set of graphs has a bounded treewidth.

**Corollary 2** *Therefore MSO queries on linguistic treebank can be evaluated in linear time in the size of the treebank.*

The above results were enhanced by several authors showing that MSO can be extended by cardinality predicates or simple counting. The perhaps most general result is given by Courcelle and Mosbah (1993) who add a certain type of evaluation functions to MSO.

The core of these results is achieved by a reduction to classical formal language theory. Using a method of semantic interpretation of one structure in another proposed by Rabin (1977), Arnborg et al. (1991), and Courcelle (1990a,b, 1992) provide a method for interpreting finite graphs of bounded treewidth by finite trees. MSO sentences can then be evaluated over these trees using tree automata techniques proposed by Doner (1970) and Thatcher and Wright (1968).

A bit more detailed, Arnborg, Lagergren, and Seese (1991) provide a general construction by which an MSO sentence  $S$  on graphs is translated into an MSO sentence  $\tau(S)$  on binary trees. This construction also transforms a general labelled graph  $G$  with a

suitable tree decomposition into a labelled binary tree  $T(G)$  in time linear in the number of vertices of  $G$  in such a way that  $S$  holds in  $G$  if and only if  $\tau(S)$  holds in  $T(G)$ . The step of the computation of a suitable tree decomposition can be done also in linear time on graphs with bounded treewidth, as was shown by Bodlaender (1996).<sup>1</sup> After the application of this transformation, classical tree automata techniques (Doner, 1970; Thatcher and Wright, 1968) can be applied.

The computation of a tree decomposition, although possible in time linear in the size of graph, is an expensive step. In a recent analysis of the original algorithm by Bodlaender (1996), Hagerup (2002) presents a variant which works three orders of magnitude faster. Still the algorithm is exponential in the square of the treewidth. This seems to indicate that it can hardly be used in practice. But there are two important facts to keep in mind that make this approach feasible. Firstly, most trees in current treebanks have a small treewidth. The capabilities of secondary relations are only sparsely used by annotators. To give an extreme example of how rarely secondary edges are used, consider the German Tübingen Treebank (Hinrichs et al., 2000), in which more than 99.9% of the trees have treewidth 1, the small rest having treewidth 2. Secondly, and more importantly, the computation of a tree decomposition has to be done only once. It is a part of the preprocessing step that transforms tree-like graphs of the input treebank into proper trees suitable for the application of tree automata techniques. Obviously, such a preprocessing is performed once and off-line. As such, it is not a relevant factor in the actual query response time, i.e., the time from the posing of a query till the presentation of the answer. Therefore, longer preprocessing times are indeed tolerable.

## 1.4 On the Expressive Power of MSO Queries

As stated in the introduction, the development of query systems that employ powerful query languages is a relatively new one. An important reason therefore is certainly the fact that corpora with rich syntactic annotations came to exist only recently. And only if there is a rich structure to query it makes sense to provide powerful query languages.

On the other hand, there is now a growing need for powerful query languages for the following reasons. Suppose a linguist is interested in finding a particular syntactic phenomenon in a large treebank. In most query languages it is a trivial task to write a query the answer set to which will contain all instances of the phenomenon that can be found in the corpus. Just write a query which is rather general. The answer set will be big and certainly contain what the linguist is looking for. But it will mainly consist of undesirable “garbage”, trees that do not exhibit the phenomenon sought. Hence, the real task in querying is not so much to produce an answer set that contains instances of what you are searching for. The task is rather to weed out the garbage, to keep answer sets as small as possible. Looking at things this way, a query is a kind of a filter for the corpus. And in order to retain small answer sets it is necessary to make that filter strong. A linguist should be able to spell out as exactly as possible the phenomenon he is looking for. And that requires powerful query languages. Treebanks have already

---

<sup>1</sup>Remarkably, Bodlaender found his result after the publication of the works by Arnborg et al. (1991) and Courcelle (1990a,b).

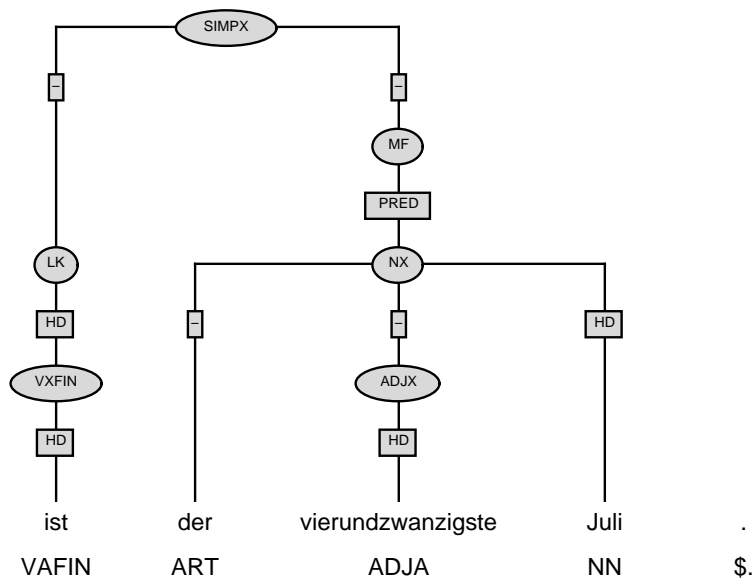


Figure 1.1: A subject-free clause from the German Tübingen Treebank

gained quite a size, e.g., the German Tübingen Treebank contains more than 38.000 trees. There is hardly any chance to manually check big answer sets any more.

Let us illustrate these arguments by linguistically motivated examples. Suppose we are looking for trees in a German or English treebank where a clause lacks the subject. It is known that Germanic languages require the subject to be lexically realised under normal circumstances. It is therefore interesting to see whether there are any exceptions from this rule, and if, what they look like. An example of a clause where the subject is missing can be seen in Figure 1.1, which displays a tree from the German Tübingen Treebank. It reads *is the twenty fourth of July*. Without going into details of the annotation we note that the grey shaded ellipses represent part-of-speech tags or syntactic categories and the grey shaded rectangles represent grammatical functions or edge labels.

In order to find trees that lack the subject, we have to find a clause node, which is a node of category (POS tag) *SIMPX* below which there is no subject. In this treebank, a subject is a node of grammatical function *ON* which stands for *Object in the Nominative*. The treebank contains a parent relation to indicate the tree structure. In our queries, we will use the dominance relation, the reflexive transitive closure of the parent relation, and designate it by the relation symbol  $\gg$ . A discussion of transitive closures in queries follows the present illustration of searching for subject-free clause.

We can now pose the following query:

$$\exists x \text{SIMPX}(x) \wedge \forall y((x \gg y) \rightarrow \neg \text{ON}(y))$$

The formula reads “There is a clause node (node of category *SIMPX*) such that no node below it is a subject node (node of function *ON* (*Object in the Nominative*)).”

An example result is the tree in Figure 1.1. Another result from the same corpus would be “*aber gut, wir können ja mal fragen, was gegeben wird.*” (*All right, we can ask,*

what's on play.) where there is no subject in the German subordinate clause. If one is interested in finding only those trees where the subject is lacking in a subordinate clause, the above query has to be extended to

$$\exists x \exists y \text{SIMPX}(x) \wedge \text{SIMPX}(y) \wedge (x \gg y) \wedge (x \neq y) \wedge (\forall z \neg((y \gg z) \wedge \text{ON}(z)))$$

*“There are two different clause nodes, one dominating the other, and no node below the lower clause node is a subject node.”*

This is a query of quantifier depth 3 (number of deepest nestings of quantifiers). On second thought one can see that this query is still too simple to find all subordinate clauses without subject. It does not reflect the possibility of having a subordinate clause with subject as a subclause of a subordinate clause without subject. Here is a query that does:

$$\begin{aligned} \exists x \exists y \text{SIMPX}(x) \wedge \text{SIMPX}(y) \wedge (x \gg y) \wedge (x \neq y) \wedge \\ (\forall z \text{ON}(z) \rightarrow (\neg(y \gg z) \vee \\ \exists w \text{SIMPX}(w) \wedge (y \gg w) \wedge (y \neq w) \wedge (w \gg z))) \end{aligned}$$

*“There are two different clause nodes, one dominating the other, and every subject node is either not dominated by the lower clause node or there is a further clause node intervening.”*

This query is even of quantifier depth 4. Another complicated query must be used if we want to find all trees in which the main clause lacks the subject, but subordinate clauses may have one. The query looks like this:

$$\begin{aligned} \exists x \text{SIMPX}(x) \wedge (\forall y \text{SIMPX}(y) \rightarrow (x \gg y \wedge x \neq y)) \wedge \\ (\forall y ((x \gg y) \wedge \text{ON}(y)) \rightarrow \\ \exists z (\text{SIMPX}(z) \wedge (x \gg z) \wedge (x \neq y) \wedge (z \gg y))) \end{aligned}$$

*“There is a highest clause node such that for every subject node dominated by it there is a second clause node intervening.”*

These examples show that once a linguist is interested in more advanced phenomena a powerful query language is necessary to specify as closely as possible what it is that the linguist seeks.

One of the advantages of MSO as a query languages is the fact, shown by Courcelle (1990a), that the transitive closure of any MSO-definable relation is also MSO-definable. Transitive closures play an important role in formal definitions of linguistic structures. Although the term is rarely literally used, many definitions contain it tacitly. One such example is the definition of dominance as given in the above discussion. Another example is the lexical head of a phrase. Here we look at the transitive closure of the head-daughter relation. Any notion of government, c-command or barriers contains indirectly a transitive closure, as well as notions of maximal or minimal categories. Since it is unforeseeable which particular variant of these notions a user querying a corpus has in mind, it is not a feasible approach to try to precompile these transitive closures during the preprocessing of the corpus. To provide a query language that allows the definition of transitive closures seems to be the more promising way.

There is a second field of applications of powerful queries, namely in the *development* of treebanks. As was pointed out by Dickinson and Meurers (2003), even treebanks

that are annotated by hand and not automatically can contain quite a number of mis-annotations and inconsistencies. To enhance the quality of the treebank an annotator can check whether his annotations are consistent by defining the environment of an annotation, querying the treebank with this definition, and inspecting if the annotations in the answer set are the way they should be. The expressive power of the query language is important for an annotator because fine-grained annotations are typically very sensitive to environments, and thus environments should be definable rather precisely.

## 1.5 Conclusion

In this paper we showed that linguistic treebanks can be queried with a very powerful query language, namely monadic second-order logic, in time linear in the size of the treebanks. We thus give an argument for that at least on a theoretical level the question of a choice of a query language for treebanks can be settled. We hardly expect the arise of a need of an even more powerful query language. And the fact that a large part of costly computations can be done in an offline preprocessing step to be performed only once lets us believe that the described approach is practically feasible.

It would certainly be nice, if one would be able to show that the types of finite structures one can find in linguistic treebanks are such that they have a bounded treewidth by their nature. But at least some of the corpus formats currently being used do not as such warrant a bound for the treewidth of its instances. A simple example is the addition of free indexation to syntax trees in GB theory such as coindexing anaphora and antecedent or moved constituents and their traces. If there is no bound on the number of coindexations, the structures have an unbounded treewidth. An inspection of some of the available treebanks reveals on the other hand, that typically only a subset of the capabilities provided by the corpus formalism is actually in use. We thus think it is an interesting research goal to see if one can find an abstract characterisation of linguistic trees as found in treebanks that is general enough to cover most existing corpora but also that specific that it provides boundedness of the treewidth of its instance structures.

## Acknowledgements

This research was funded by a grant of the German Science Foundation (DFG SFB 441-2). I would like to thank Uwe Mönnich for interesting and helpful discussions.

## Bibliography

- Abeillé, A. and L. Clément (1999). A tagged reference corpus for French. In *Proceedings of EACL-LINC*.
- Arnborg, S., J. Lagergren, and D. Seese (1991). Easy problems for tree-decomposable graphs. *Journal of Algorithms*, **12**:308–340.

- Bodlaender, H. L. (1993). A tourist guide through treewidth. *Acta Cybernetica*, **11**:1–23.
- Bodlaender, H. L. (1996). A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, **25**:1305–1317.
- Brants, S., S. Dipper, S. Hansen, W. Lezius, and G. Smith (2002). The TIGER Treebank. In K. Simov, ed., *Proceedings of the Workshop on Treebanks and Linguistic Theories*. Sozopol.
- Brants, T., W. Skut, and H. Uszkoreit (1999). Syntactic annotation of a German newspaper corpus. In *Proceedings of the ATALA Treebank Workshop*, pp. 69–76.
- Courcelle, B. (1990a). Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, volume B, chapter 5, pp. 193–242. Elsevier.
- Courcelle, B. (1990b). The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, **85**:12–75.
- Courcelle, B. (1992). The monadic second-order logic of graphs III: Tree-decompositions, minors and complexity issues. *Informatique Théorique et Applications*, **26**:257–286.
- Courcelle, B. and M. Mosbah (1993). Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science*, **109**:49–82.
- Dickinson, M. and D. Meurers (2003). Detecting errors in part-of-speech annotations. In A. Copestake and J. Hajič, eds., *Proceedings EACL 2003*, pp. 107–114.
- Doner, J. (1970). Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, **4**:406–451.
- Ebbinghaus, H.-D. and J. Flum (1995). *Finite Model Theory*. Springer-Verlag.
- Hagerup, T. (2002). Simpler and faster tree decomposition. Manuscript, University of Frankfurt a. M.
- Hinrichs, E., J. Bartels, Y. Kawata, V. Kordoni, and H. Telljohann (2000). The VERB-MOBIL treebanks. In *Proceedings of KONVENS 2000*.
- Kallmeyer, L. and I. Steiner (2002). Querying treebanks of spontaneous speech with VIQTORYA. *Traitement Automatique des Langues*, **43**(3):155–179.
- Kepser, S. (2003). Finite Structure Query: A tool for querying syntactically annotated corpora. In A. Copestake and J. Hajič, eds., *Proceedings EACL 2003*, pp. 179–186.
- König, E. and W. Lezius (2000). A description language for syntactically annotated corpora. In *Proceedings of the COLING Conference*, pp. 1056–1060.
- Marcus, M., B. Santorini, and M. A. Marcinkiewicz (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, **19**(2):313–330.

- Rabin, M. (1977). Decidable theories. In J. Barwise, ed., *Handbook of Mathematical Logic*, pp. 595–629. North-Holland.
- Randall, B. (2000). CorpusSearch user’s manual. Technical report, University of Pennsylvania. <http://www.ling.upenn.edu/mideng/ppcme2dir/>.
- Robertson, N. and P. Seymour (1986). Graph minors II. Algorithmic aspects of treewidth. *Journal of Algorithms*, **7**(309–322).
- Rohde, D. (2001). Tgrep2. Technical report, Carnegie Mellon University. <http://tedlab.mit.edu/~dr/Tgrep2/>.
- Thatcher, J. and J. Wright (1968). Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, **2**(1):57–81.
- Vardi, M. (1982). The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing*, pp. 137–146.
- Wallis, S. and G. Nelson (2000). Exploiting fuzzy tree fragment queries in the investigation of parsed corpora. *Literary and Linguistic Computing*, **15**(3):339–361.