17

Linguistic Grammars with Very Low Complexity

ANSSI YLI-JYRÄ

17.1 Introduction

Fifteen years ago, in the COLING-90 in Helsinki, Kimmo Koskenniemi sketched a finite-state approach to surface syntax (Koskenniemi, 1990): an approach that later became known by the name Finite-State Intersection Grammar (FSIG). During the subsequent few years this approach was investigated by Koskenniemi's associates Pasi Tapanainen, Atro Voutilainen and some others in the Research Unit of Multilingual Language Technology at the Department of General Linguistics at the University of Helsinki.

A while after Koskenniemi's proposal, technical problems related to the state complexity of FSIG grammars became a major challenge in the further development of the system. However, this was largely due to the fact that the rules in the first grammars did not suggest means to exploit the locality of linguistic constraints. Meanwhile, a similar but less ambitious constraint system flourished independently in France as Maurice Gross and his students had introduced *local grammars* and developed algorithms that apply these to lexically ambiguous sentences.

In 1995, the current author became involved, for the first time, in investigations that pursued more efficient FSIG parsing strategies and complexity. These investigations continued in 2000's and led to a PhD thesis. This chapter tries to give an overview of the recent discoveries related to the complexity of FSIG parsing. The chapter is structured as follows: Section 17.2 sketches a rough background of Kimmo's approach, relating FSIG to the well known CG framework, and to finite-state methods in general. Section 17.3 general-

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

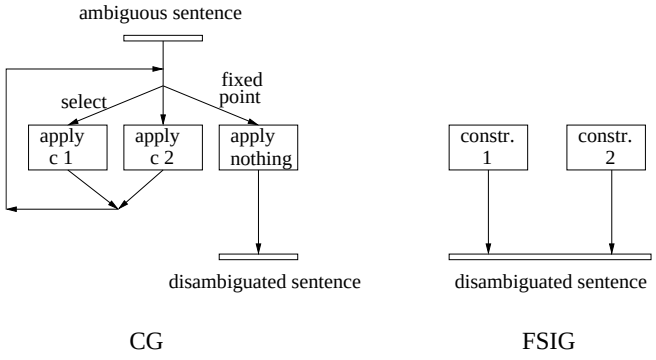

FIGURE 1 FSIG and its forerunner: Constraint Grammar.

izes the FSIG architecture to non-regular languages and linguistically important structures. Section 17.4 explains the star-freeness property of the FSIG grammars. Section 17.5 approaches FSIG parsing through a layered structure, essentially improving the compactness of the grammar.

17.2 The Background of FSIG

17.2.1 The Inspiration

Kimmo's FSIG approach was largely inspired by the Constraint Grammar (CG) system (Karlsson, 1990) that can be described as follows.

The input of CG is a tokenized sentence with alternative readings listed at each token. Each CG constraint rule application is a transformation that removes one or more readings of an ambiguous token in a given context. Each transformation is a rational transduction. The context conditions tested by the rules are able to refer to contextual ambiguity and to test bunches of alternative readings in the token and its context.

As a whole, a CG parser is a combination of a prioritized union of rules that is applied iteratively up to a fixed point where no rule can reduce any more ambiguity. The parser works by iteratively selecting a constraint and an ambiguous token and applying the selected constraint to the selected token. This process terminates: the maximal number of iterations carried out by the parser is proportional to the length of the sentence. Furthermore, the order in which the tokens are processed may require a linear number of back and forth jumps between token positions. It is thus not generally possible to characterize a CG parser (Figure 1, left side) as a regular relation.

In 1983, Kimmo Koskenniemi had became an inventor of a parallel constraint system, the two-level model of morphology (Koskenniemi, 1983) (TWOL). TWOL had already been proved a practical alternative to a cascade

of phonological rules, such as used in generative phonology. As a natural continuation to this work, Kimmo proposed a similar approach to syntactic parsing and disambiguation: a system of parallel finite-state constraints that defined a regular relation (Koskenniemi, 1990). This system presented an alternative for the serial approach of the CG parsing, but was not claimed to be equivalent to it.

17.2.2 FSIG as a One-Level System

The formal elegance of FSIG was remarkable. It was a one-level system, while, in contrast, the number of intermediate levels in CG was not bounded by a constant. Furthermore, FSIG was able to give new insight on possible parsing approaches by putting into practice a set-theoretic semantics for grammars. FSIG parsing consists of two phases:

1. generation of a set of potential reading strings, and
2. constraint-driven selection of the grammatical readings.

First, potential *sentence readings* of the input sentence — each being a string of morphemes and word boundaries — are constructed by inserting annotation codes freely into the sequence of input tokens (in practice, it is desirable that the insertion is controlled by lexical lookup). This creates a set of alternatives that is often referred to as an *ambiguous sentence*[1]. The generated set is represented by a deterministic finite automaton (DFA) that is often called the *sentence automaton*[2].

Second, there are *constraint automata* each of which implements a linguistic or administrative constraint, originally described using an extended FSIG notation of regular expressions. When a new ambiguous sentence has been generated the constraint automata are applied to reduce ungrammatical sentence readings (strings) in the sentence automaton. This can be carried out, in theory, by computing a direct product of the sentence automaton and the constraint automata, or by performing a backtracking search for alternative analyses. The direct product automaton describes exactly those sentence readings that are recognized by every constraint automaton.

In contrast to the fixed point semantics of the CG disambiguation process, the standard FSIG is a one-level constraint system with hard constraints: if some constraint rejects all potential readings, there will be no readings left in the output. An alternative FSIG framework with soft constraints would be desirable for purposes of robust parsing (although we do not want to run into the practical difficulties of Optimality Theoretic approaches).

[1] Generation of this set resembles the GEN function in Optimality Theory, but can be already constrained by the lexicon.

[2] Earlier, this automaton used to be acyclic, but this restriction is no more maintained in recent FSIG systems.

17.2.3 FSIG and State Complexity

In addition to the parallel constraints, the initial developers of the FSIG framework adopted a useful operation from the formalism of the original two-level morphology: the so-called context restriction operator became a part of the FSIG notation. This regular operator has an interesting history, and it allows for further generalizations (Yli-Jyrä and Koskenniemi, 2004).

The FSIG rule formalism is able to specify complex finite-state grammars in a very compact fashion. In fact, we could estimate that the deterministic state complexity (the size of the minimal DFA) corresponding to the combination of all constraint automata of a full-coverage grammar could be some 10^{10} – 10^{1000} states. The constraints can be applied in linear time to the input sentence, according to the the input size, but linear time complexity alone does not thus imply a practical implementation.

It is also important to understand how the complexity of the grammar is related to the way the grammar is designed. A few initial results on state complexity of *bracketing restriction operator* – a recent novelty (Yli-Jyrä, 2003c) – and *context restriction operator* (Yli-Jyrä and Koskenniemi, 2004) have been published. Their complexity grows, in the worst case, exponentially according to the maximum depth of bracketing.

17.2.4 The Quest for Locality in FSIG

Some FSIG experts, including Kimmo himself, have always maintained the optimism that an efficient parser for FSIG could be found. The hope is motivated by the fact that the parse result – the reduced set of alternatives – does not exhibit remarkable state complexity although its computation is difficult. To be more successful, an efficient parser would need to decompose the grammar in a fashion that maintains compactness during the intermediate parsing steps. How this should be done has been an open problem, but a recently presented compilation method for rules (Yli-Jyrä and Koskenniemi, 2004) sheds some light on how the grammar can be split into almost independent modules.

An informal comparison to a personal computer may be helpful in understanding compact representations of finite automata and transition functions: CPUs implement predefined state transitions in an immense state space, without any difficulties. This is possible because (i) the CPUs modify, within one step, only a small portion of the computer's memory, making only *local state transitions* at a time, and (ii) the next state is often computed in parallel, largely independent circuits within the processor (such as the program counter, the arithmetic logic unit and the cache). If similar design principles – locality and decomposition – could be used to store the FSIG grammar and the intermediate results, we could perhaps find an efficient parser.

17.3 A generalization of Kimmo's Approach

17.3.1 Anti-Approximations

Constraint grammars (CGs) and local grammars are typically applied to a flat sequence of lemmas and tags, without any attempts to cope with bracketed trees. In contrast to this, Kimmo's proposal and the first FSIGs included clausal embedding up to one level of clause boundaries. It was argued by Kimmo that only a tiny proportion of running-text sentences would contain a double-center-embedded clause.

The current author (Yli-Jyrä, 2003a) considered explicitly an arbitrary limit d for center embedding in FSIG. This generalization suggests the possibility of taking union of the languages of an FSIG grammars G when its d-parameter goes to infinite:

$$L(\hat{G}) = \cup_{d=0..\infty} L(G_d) = \lim_{d\to\infty} L(G_d).$$

According to the formula, every FSIG grammar G, such as Voutilainen's English grammar (Voutilainen, 1997) is in fact a parameterized specification that gives us both

- a series of finite-state grammars $G_0, G_1, G_2, \ldots$, and
- an idealistic generalization, an *anti-approximation* $L(\hat{G})$.

In the case of Voutilainen's English grammar, the anti-approximation $L(\hat{G})$ is context-free, but, in some other cases, it can be non-context-free[3].

This view suggests a perspective on how non-regular grammars could be learned: through a series of regular languages. Furthermore, the view suggests connections to bracket-based representations of non-regular grammars.

17.3.2 Chomsky-Schützenberger Representations

In early 1960's, Noam Chomsky and Marcel Paul Schützenberger (1963) discovered a technique to represent the language of any context-free grammar G as a homomorphic image of an intersection of a Dyck language and a regular language that depends on G.

FSIG grammars have a close relationship to the Chomsky-Schützenberger representations. In any FSIG grammar G_d, the parameter d actually specifies an approximation of a Dyck language. By using, in the constraint semantics, a Dyck language instead of its regular approximation, we get a representation for the anti-approximated language $L(\hat{G})$.

This view has been very fruitful. We have been able to specify many new Chomsky-Schützenberger style representations[4] for various classes of

[3] This is possible if the grammar uses crossing sets of brackets as in (Yli-Jyrä, 2004).

[4] In some of our representations, the Dyck language is distributed and used as a constant in the rules. We refer to them loosely as Chomsky-Schützenberger style representations.

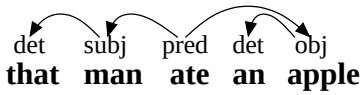

FIGURE 2 A dependency tree.

formal grammars. There are such representations for (i) extended context-free languages, (ii) projective dependency grammars, and (iii) certain mildly context-sensitive grammars (MCSGs) that correspond to some families of non-projective dependency grammars.

A tantalizing opportunity of this approach is to try and develop similar bracketing-based representations to further examples of MCSGs, such as tree-adjoining grammars and (multi-modal) combinatorial categorial grammars. Whether this can be done is an open problem, but a success would greatly increase the relevance of the FSIG framework to Natural Language Processing (NLP), since a single architecture would allow for both an idealistic generalization and a series of finite-state approximations.

17.3.3 A New Bracketed Representation for Dependencies

According to Koskenniemi (1990), his approach *does not aim to uncover semantically oriented distinctions.* This limitation was maintained in the first FSIG systems that were clearly meant for partial parsing and not for, *e.g.* producing an explicit dependency structure.

Dependency links indicate, ideally, how words with predicate-argument structures are composed in a semantically coherent way. As we saw in the above, recent developments of FSIG have introduced new sub-frameworks, including frameworks for projective and non-projective dependency parsing.

For example, the dependency tree in Figure 2 can be represented as a bracketed string as in Figure 3.

#			**that**	det	$[_{\leftarrow}$
#	$]_{\leftarrow}$		**man**	subj	$[_{\leftarrow}$
#	$]_{\leftarrow}$		**ate**	pred	$[_{\rightarrow}$
#			**an**	det	$[_{\leftarrow}$
#	$]_{\leftarrow}$	$]_{\rightarrow}$	**apple**	obj	#

FIGURE 3 String with dependency-tree bracketing. The line breaks have been added.

In non-projective dependency structures (Yli-Jyrä, 2003b, 2004), we use disjoint sets of brackets and follow a non-trivial generalization of the so-called stack discipline when allocating crossing brackets. Thanks to this, each non-projective structure has a unique encoding as a bracketed string. The corresponding system of stacks is connected to a class of MCSGs (Yli-Jyrä and

Nykänen, 2004).

The ability of recent FSIGs to cope with dependency trees (and graphs) under certain performance restrictions suggests that the framework might be capable of assigning even some semantically coherent structures in terms of syntactic dependencies.

17.4 A Characterization of the Complexity of FSIG

17.4.1 Background

The *star-free languages* are the smallest class of languages that contains all finite languages and is closed under concatenation and the Boolean operations.

In Coding Theory, Schützenberger (1965) made a seminal finding by characterizing the star-free languages with aperiodic finite syntactic monoids[5]. Mathematics of Coding Theory and in particular the study of star-free languages are inherently connected to linguistic performance, communication and error tolerance, but star-free languages are seldom discussed in linguistic literature. As a positive example, Kornai (1985) argues on practical limits in natural language semantics, and how this supports an assumption of star-freeness of natural language. The relevance of star-free languages to the language acquisition task has been demonstrated separately by learning algorithms that cope with certain star-free classes of regular languages (Segarra et al., 2003).

17.4.2 Establishment of Star-Free FSIGs

The property of star-freeness has been recently assigned to the FSIG framework. First, the star-freeness of the annotated language described by Voutilainen's English FSIG was established through a rewriting approach (Yli-Jyrä, 2003a). Second, it has become increasingly clearer that the star-freeness restriction does not imply essential losses in the linguistic applicability of FSIGs although it is not difficult to construct artificial examples of FSIGs that fail to be star-free. This is indicated by the flavors of star-free FSIGs that coped with various syntactic structures, including bracketed string representations for unranked constituent trees, projective dependency trees and restricted non-projective dependency trees.

17.4.3 Definability in the First-Order Logic

Robert McNaughton and Seymour Papert (1971) discovered that star-free languages are exactly those described in $FO[<]$, a fragment of first-order logic whose signature contains linear order relation $<$ over string positions. This result is important because it connects star-free languages, such as described

[5]Transitions of a minimal DFA $A = (Q, i, F, \Sigma, \delta)$ define for $w \in \Sigma^*$ the function $\delta_w : Q \to Q$. The set of all functions δ_w with a composition operator and the identity element δ_ϵ is the transition monoid of A as well as the syntactic monoid of the language $L(A)$.

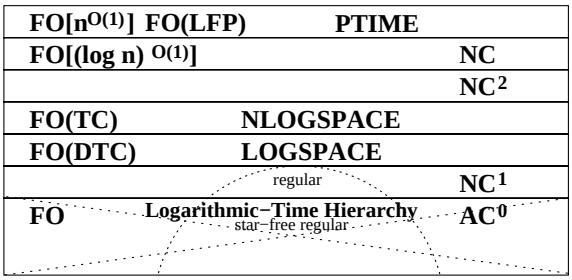

FIGURE 4 Computational complexity of polynomial-time problems, adapted from Immerman (1999).

by FSIG grammars, to (i) the locality characterizations of first-order definable structures, and (ii) to descriptive complexity.

First, we get access to a famous theorem by W. Hanf (Immerman, 1999, p.102-103). According to this theorem, first-order formulae with a *bounded quantifier rank*[6] cannot distinguish between two graphs of bounded degree if the graphs have the same number of local neighborhoods of all possible types where the number of possible types depends exponentially on the quantifier rank. The definition of locality is here more general than in NLP since it involves quantification.

Second, we get access to results in Finite Model Theory, where many computational complexity classes have been characterized using fragments of first-order logic. The close relationship between the computational complexity of problems and the richness of logical language needed to describe them — their descriptive complexity — was established when Ron Fagin showed in 1974 that the problems computable in nondeterministic polynomial time (NP) are exactly characterized by the problems that can be described in existential second-order logic. Neil Immerman (1999, p.2) summarizes the role of descriptive complexity as follows:

> *It [descriptive complexity] gives a mathematical structure with which to view and set to work on what had previously been engineering questions.*

17.4.4 Parallel Computational Complexity

When the languages definable with $FO[<]$ are placed into the picture of computational complexity classes, we observe that they correspond, as illustrated in Figure 4,

- to the logarithmic-time hierarchy, and
- to the uniform circuit complexity class AC^0.

[6]The quantifier rank of a first-order formula is basically the number of nested quantifiers.

A short explanation for some classes in Figure 4 is in place. The *logarithmic-time hierarchy* (LH) contains languages that can be recognized with an alternating Turing machine (ATM) in *logarithmic time* (according to the length of input) using a bounded number of alternations between existential and universal states. The *circuit complexity class* AC^0 consists languages whose strings can be recognized using a constant depth, unbounded-fan-in polynomial-size AND-OR circuits. The circuits in the class NC^1 differ from AC^0 by having a logarithmic depth according to the length of the strings, but restricting the AND and OR gates to ones with two fan-ins.

Star-freeness implies an essential restriction to the parallel computational complexity and circuit complexity of regular languages. Among all regular languages, there are some that do not belong to AC^0, but all are included in NC^1. AC^0 contains all star-free regular languages (Thomas, 1997).

17.5 Structure of Annotated FSIG Languages

17.5.1 The Dot-Depth Hierarchy

Based on the star-freeness of FSIGs, we are able to study the means to represent and parse these grammars in a compact fashion. Star-free languages admit representations that are not available to regular languages in general.

A particularly interesting representation of star-free languages is based on the closure of finite languages, Boolean operations and so-called *concatenation products*. Such a representation of star-free languages generates an infinite sequence or hierarchy of language classes. One of the possible sequences is the dot-depth hierarchy $\mathbf{B}_0, \mathbf{B}_1, \mathbf{B}_2, \ldots$ that was introduced by Brzozowski and Knast (1978). It defines the set of all star-free languages:

$$SF = \cup_{i=0\ldots\infty}\mathbf{B}_i = \lim_{i\to\infty} \mathbf{B}_i.$$

The dot-depth hierarchy is defined over an alphabet Σ as follows:

- $\mathbf{B}_0$ consists of finite and co-finite subsets of Σ^*,
- $\mathbf{C}_i$ consists of concatenations of languages in $\mathbf{B}_i$,
- $\mathbf{B}_i$ consists of Boolean combinations of languages in $\mathbf{C}_{i-1}$.

Thomas (1982) showed that the dot-depth hierarchy corresponds to the quantifier-alternation and logarithmic-time hierarchies mentioned above. According to Thomas, language $L \in \mathbf{B}_i$ can be described by a prenex normal-form that has a so-called Σ_i prefix of quantifiers[7].

If we knew the lowest dot-depth level $\mathbf{B}_i$ that contains the language (the set of annotated strings) of a grammar, we could say more about the parallel computational complexity of the grammar. Unfortunately, the determination

[7] A formula is in prenex normal-form if it consists of a string of quantifiers applied to a quantifier free formula.

of the exact dot-depth of a language is a difficult (open) problem. We can still easily approximate the level from above because the dot-depth depends mainly on d, the depth of allowed bracketing. For example, approximations of the Dyck language can be constructed using a recursive star-free formula (Yli-Jyrä, 2003a) that specifies a language belonging to $\mathbf{B}_{O(d)}$. This can be shown easily by the structure of the recursive formula.

In summary, understanding of the locality in FSIG grammars can be largely built around the relationship between descriptive and parallel computational complexity, d, the dot-depth and the size of the quantifier prefix.

17.5.2 Relative State Complexity of FSIGs

Parallel computational complexity of FSIGs is not just about nondeterministic time. If the minimal DFA equivalent to an ATM could be constructed in a straightforward way (the problem is undecidable for arbitrary ATMs), regular operations applied during the construction would contribute to the *state complexity* of the result. It is imaginable that the alternation between the existential and universal states would amount for additional steps in the state complexity.

In an FSIG approximation of non-projective dependency grammars, the depth d of bracketing corresponds to the number of alternations between concatenations and Boolean operations, while the combination of n disjoint sets of brackets corresponds to an intersection of n star-free languages. Both of these parameters are able to cause an exponential growth in the state complexity of some pathological FSIG grammars.

In the structure of FSIG languages, certain language class distinctions seem to differ radically from the Chomsky hierarchy. For example, if we anti-approximate the mentioned FSIG implementations of non-projective dependency grammars, n induces a hierarchy of MCSGs. Such hierarchies of MCSGs are often seen to exhibit a competence feature, while limited clausal embedding would be an example of a performance feature. In contrast to this complexity landscape, both these parameters (n crossings and d embeddings) appear to be equally important when we determine the state complexity of an FSIG, and the number of crossing sets of brackets (n) does not have any role, when we determine the asymptotic bound for the dot-depth.

17.5.3 Parallel Decompositions

Each new dot-depth level makes references to lower dot-depth levels in a similar fashion as compact parse forest representations pack ambiguity that is beyond the domain of locality. This observation suggests a compact parallel representation for FSIG grammars where $n = 1$ and $d > 0$ (an FSIG with $n > 1$ is obtained by combining simpler FSIGs under intersection). The use of such a representation requires the following steps:

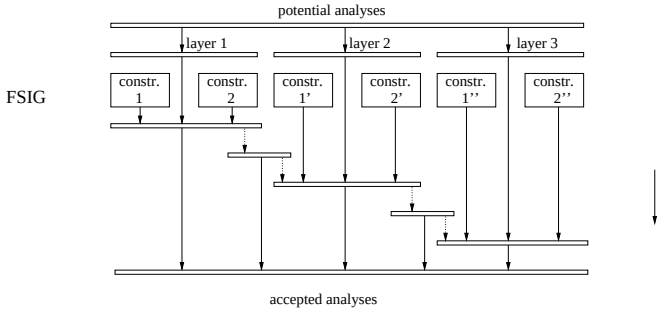

FIGURE 5 FSIG parsing with layers and sub-grammars.

- decomposition of each FSIG constraint into separate constraints each of which checks one layer, *i.e.* level of brackets (>sub-grammars),
- applying the constraints of each layer into a separate copy of the sentence automaton, and
- combining all the constrained sentence automata to obtain the final result.

These tasks have been discussed more in detail by the author (Yli-Jyrä, 2005). A rough overview of the proposed parsing strategy is presented in Figure 5.

17.6 Conclusion

We have presented an overview of the FSIG approach and related FSIG grammars to issues of very low complexity and parsing strategy. We ended up with serious optimism according to which most FSIG grammars could be decomposed in a reasonable way and then processed efficiently.

References

Brzozowski, Janusz A. and Robert Knast. 1978. The dot-depth hierarchy of star-free languages is infinite. *Journal of Computer and System Sciences* 16:37–55.

Chomsky, Noam and Marcel-Paul Schützenberger. 1963. The algebraic theory of context-free languages. In P. Braffort and D. Hirschberg, eds., *Computer Programming and Formal Systems*, pages 118–161. Amsterdam: North Holland.

Immerman, Neil. 1999. *Descriptive Complexity*. Graduate Texts in Computer Science. New York: Springer-Verlag.

Karlsson, Fred. 1990. Constraint grammar as a framework for parsing running text. In H. Karlgren, ed., *13th COLING 1990, Proceedings of the Conference*, vol. 3, pages 168–173. Helsinki, Finland.

Kornai, András. 1985. Natural language and the Chomsky hierarchy. In *2nd EACL 1985, Proceedings of the Conference*, pages 1–7. Geneva, Switzerland.

Koskenniemi, Kimmo. 1983. *Two-level morphology: a general computational model for word-form recognition and production*. No. 11 in Publications of the Department of General Linguistics, University of Helsinki. Helsinki: Yliopistopaino.

Koskenniemi, Kimmo. 1990. Finite-state parsing and disambiguation. In H. Karlgren, ed., *13th COLING 1990, Proceedings of the Conference*, vol. 2, pages 229–232. Helsinki, Finland.

McNaughton, Robert and Seymour Papert. 1971. *Counter-Free Automata*. No. 65 in Research Monograph. Cambridge, Massachusetts: MIT Press.

Schützenberger, Marcel Paul. 1965. On finite monoids having only trivial subgroups. *Information and Computation (Information and Control)* 8(2):190–194.

Segarra, E., E. Sanchis, F. García, L. Hurtado, and I. Galiano. 2003. Achieving full coverage of automatically learnt finite-state models. In *Proceedings of the Workshop on Finite-State Methods in Natural Language Processing*, pages 135–142. Agro Hotel, Budapest.

Thomas, Wolfgang. 1982. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences* 25:360–376.

Thomas, Wolfgang. 1997. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages*, pages 389–455. Springer.

Voutilainen, Atro. 1997. Designing a (finite-state) parsing grammar. In E. Roche and Y. Schabes, eds., *Finite-State Language Processing*, chap. 9, pages 283–310. Cambridge, MA, USA: A Bradford Book, the MIT Press.

Yli-Jyrä, Anssi Mikael. 2003a. Describing syntax with star-free regular expressions. In *11th EACL 2003, Proceedings of the Conference*, pages 379–386. Agro Hotel, Budapest, Hungary.

Yli-Jyrä, Anssi Mikael. 2003b. Multiplanarity – a model for dependency structures in treebanks. In J. Nivre and E. Hinrichs, eds., *TLT 2003. Proceedings of the Second Workshop on Treebanks and Linguistic Theories*, vol. 9 of *Mathematical Modelling in Physics, Engineering and Cognitive Sciences*, pages 189–200. Växjö, Sweden: Växjö University Press.

Yli-Jyrä, Anssi Mikael. 2003c. Regular approximations through labeled bracketing (revised version). Submitted to the post-proceedings of the 8th Formal Grammar conference, Vienna, Austria, 16–17 August, 2003. A link available at `http://www.ling.helsinki.fi/~aylijyra/dissertation/`.

Yli-Jyrä, Anssi Mikael. 2004. Axiomatization of restricted non-projective dependency trees through finite-state constraints that analyse crossing bracketings. In G.-J. M. Kruijff and D. Duchier, eds., *Proc. Workshop of Recent Advances in Dependency Grammar*, pages 33–40. Geneva, Switzerland.

Yli-Jyrä, Anssi Mikael and Kimmo Koskenniemi. 2004. Compiling contextual restrictions on strings into finite-state automata. In L. Cleophas and B. W. Watson, eds., *The Eindhoven FASTAR Days, Proceedings*, no. 04/40 in Computer Science Reports. Eindhoven, The Netherlands: Technische Universiteit Eindhoven.

Yli-Jyrä, Anssi Mikael and Matti Nykänen. 2004. A hierarchy of mildly context sensitive dependency grammars. In G. P. Gerhard Jäger, Paola Monachesi and S. Wintner, eds., *Proceedings of the 9th conference on Formal Grammar 2003 "FGNancy"*.

Yli-Jyrä, Anssi Mikael. 2005. *Contributions to the Theory of Finite-State Based Linguistic Grammars*. No. 38 in Publications of the Department of General Linguistics, University of Helsinki. Helsinki: Helsinki University Printing House.