1. Background

My Ph.D. project in the Department of General Linguistics at the University of Helsinki started in 2002, under the title "Efficient parsing with finite-state constraint satisfaction". The research concerns a specific finite-state method for parsing, and its goal is to increase practical value and flexibility of the method. In this article, I present a streamlined version of my Ph.D. research plan.

I have studied Computational Linguistics (major) and Computer Science. Before my Ph.D. project, I worked in two long projects. During 1992-1997 I was involved in a parallel corpus project which yielded a detailed multilingual word index. In 1998–2001 I developed various tools for Web indexing. However, I have always been interested in finite-state methods because of their conceptual clarity. Finite-state automata are efficient, and therefore a widely employed in human language technology. Already in 1995, the example of my colleague Jussi Piitulainen (Jussi Piitulainen: "Locally tree-shaped sentence automata and resolution of ambiguity", NODALIDA-95, 1995) greatly inspired me, and I started to investigate finite-state methods.
2. Introduction

Kimmo Koskenniemi has proposed a parsing method that uses finite-state automata as a descriptive model (Kimmo Koskenniemi: "Finite-state parsing and disambiguation", COLING'90, 1990). This method is usually referred to as the Finite-State Intersection Grammar (FSIG) framework.

FSIG is related to a more widely known framework of Constraint Grammar (Fred Karlsson: "Constraint grammar as a framework for parsing running text", COLING'90, 1990). The grammar rules of FSIG work parallel, while in CG they take actions one after another. A CG rule can drop less probable readings for words, but an FSIG rule says definitely how a correct analysis for the sentence should look like.

Although finite-state methods are usually associated with efficient implementations, this has not seemed to be true in the case of FSIG. The current implementations of FSIG parsers are sometimes extremely slow although the parser works in asymptotic linear time. My purpose is to examine how the FSIG framework can be improved by introducing new finite-state methods that increase flexibility of the data-structures. In the following I have divided my project into three subgoals.

3. Compact representation

My first subgoal is to find an extremely compact representation for the FSIG description during the parsing process. In fact, the main reason for slowness is that the finite automaton that is implicitly used in the method is huge, in fact too big to fit (as a deterministic automaton) into any computer. So, even the linear time complexity in terms of the size of input grammar and sentence is prohibitive.

By using a compact representation, the blow-up of the automaton can be avoided and the FSIG parsing problem can be completely solved in many natural cases (see e.g. Yli-Jyrä: “Structural correspondence between Finite-State Intersection Grammar and Constraint Satisfaction Problem”, FSMNLP 2001, 2001).

I my first FSIG parser, I introduced a simple technique that is useful in solving the parsing problem (Yli-Jyrä: "Schematic finite-state intersection parsing", NODALIDA-95, 1995). Yet, my parser was only a bit faster (for very long sentences) than a test implementation provided by Pasi Tapanainen, who had experienced with a collection of various strategies and algorithms (Pasi
Tapanainen: "Applying a finite-state intersection grammar", in "Finite-state language processing", edited by E. Roche and Y. Schabes, 1997). In my Ph.D. project I started to work on the same technique again. This time, however, I study various ways in which my original technique can be combined with other efficient data structures that are well-known in other grammar frameworks.

In order to adopt efficient data structures to finite-state parsing, I will investigate the relationship between FSIG and some other grammar frameworks. For this purpose, it is not enough that we can place the grammar systems into the Chomskian hierarchy of language types. We should also be aware of the similarities between the different models of syntax.

4. Model-Theoretic Syntax

My second subgoal is to relate the FSIG framework with other grammar frameworks and to determine conditions under which different grammar formalisms are interchangeable tools. In other words, I try to bridge the gap between Koskenniemi’s framework and some other grammar systems. In practice, this can be approached by studying finite-state representations for ambiguous context-free trees during parsing, or it is also possible to resort to a logical language in order to relate grammars that are based on different perspectives.

This subgoal has also direct practical value. It is important that we can choose the best-suited grammar formalism for each task. However, linguists using various syntactic formalisms should have a common interlingua that can be used in sharing grammar fragments. One of the most striking oppositions exists between generative and reductionistic approaches which seem to have nothing in common. If we could write a syntactic description by combining different grammar writing "styles", development of parsers would become easier.

5. Practical implementation and experiments

My final subgoal is to demonstrate that FSIG parsing provides a practical implementation. This involves carrying out some experiments with various parsing algorithms and so it naturally means working with an implementation of my own.
Although it is very time-consuming to write implementations along with the research, a lot of programming is unavoidable. For historical and technical reasons, I have chosen C as the primary programming language. In order to minimize the extra work I plan to use, as far as possible, existing FSIG descriptions and test suites (Atro Voutilainen: "Designing a (finite-state) parsing grammar", in "Finite-state language processing", edited by E. Roche and Y. Schabes, 1997).

The software development will be based on my earlier parser implementation. In the summer 2002, I have reorganized and adapted the software for teaching purposes, with GNU licensing in my mind.

6. Summary of the first year

During 2002, I have studied the relationship between logic and automata. Moreover, I have studied new parsing strategies for FSIG descriptions. The results have been promising and they have led to an efficient solution for certain restricted grammars.

I have been working on two unfinished reports on the special cases of FSIG descriptions. In the spring 2002, I also spent several weeks preparing material for my course on finite-state automata. In May 2002, I gave my course on finite-state automata for the first time. The material (including the software) has been prepared so that it can be used in a possible course within Nordic cooperation programs during 2003.