Contents

1 Extended finite state models of language

ANDRÁS KORNAI

In spite of the wide availability of more powerful (context free, mildly context sensitive, and even Turing-equivalent) formalisms, the bulk of the applied work on language and sublanguage modeling, especially for the purposes of recognition and topic search, is still performed by various finite state methods. In fact, the use of such methods in research labs as well as in applied work actually increased in the past five years. To bring together those developing and using extended finite state methods to text analysis, speech/OCR language modeling, and related CL and NLP tasks with those in AI and CS interested in analyzing and possibly extending the domain of finite state algorithms, a workshop was held in August 1996 in Budapest as part of the European Conference on Artificial Intelligence (ECAI'96).

The present volume in the ACL Studies in Natural Language Processing series grew out of the proceedings of this workshop, available in prepublication format from the von Neumann Society of Computer Science (Báthori u. 16, H-1054 Budapest, Hungary) in hard copy (see also `www.cs.rice.edu/~andras/ecai.html`), and in a more polished but much shorter version as a special issue (Vol 2, No. 4) of **Natural Language Engineering**. Readers of this volume are advised to look at these versions, since they contain several excellent articles not included in the volume because the authors felt that their subsequent work took a direction such that they no longer consider the workshop paper fully representative of their current thinking or simply because the editor wanted to minimize overlap.

The volume is accompanied by a cd-rom containing six subdirectories: `ECAI`, `Kanungo`, `Kim`, `Kornai`, `Uniparse`, and `Watson`. From `ECAI` we call attention to the tutorial paper by Jelinek (excerpted from his book (Jelinek 1997)); the paper by Oehrle on binding and anaphora; and the various commentaries presented at the workshop. Of the NLE special issue (not available on the cd-rom), we call attention to the paper by Abney on finite state cascades, the Karttunen *et al.* paper describing recent developments in the Xerox finite state calculus, the paper by Koskenniemi on morphological problems arising in the context of information retreival, and the paper by Sproat describing the application of weighted transducers in text-to-speech systems. Size limitations of the NLE special issue made it impossible to include more than a brief abstract of some papers there. The full version of these papers, taking into account the comments received

at the workshop, appears in this volume for the first time. In addition, a formal Call for Papers yielded several new papers for this volume, making the original proceedings, the NLE special issue, and the present volume independently valuable for researchers in this area.

By including the original `ECAI` papers, the cd-rom comes closer to reflecting the true breadth of research in finite state language modeling than the source code contributions would suggest. The problem is, of course, that most of the authors represented in the volume make a living by building finite state models and are not in a position to give out source code, which is usually treated as highly proprietary to the companies where they work. Were it not for special circumstances, namely that the company where Kornai did the work sank without a trace, and the company where Watson works is willing to see old versions published (an enlightened attitude the whole software industry would do well to emulate), only the `Kanungo, Kim,` and the `Uniparse` directories, written in an academic environment, could be published here.

The reader mounting the cd-rom will not find the neatly packaged distributions that are now standard in the world of free software. To make use of the software presented here requires more than just running a `make` – it requires serious programming work. While this is a unix distribution through and through, no sophisticated system calls are used, so the bulk of the underlying code is trivially PC-able (the few `fork(2)` calls in `Uniparse` are a possible exception), and the inclusion of several megabytes of newspaper text and other English lexical resources should also help in bringing the solitary developer closer to the mainstream. To be sure, the `Kanungo` system is rather skeletal when compared to major Hidden Markov systems like HTK (see `http://www.entropic.com/support/FAQ/FAQ_htk.html`), and the `Watson` toolkits do not nearly have the convenience of XFST (see `http://www.rxrc.xerox.com/research/mltt/fsSoft/docs/fst-97/xfst97.html`). But for someone interested in gaining a practical understanding of the main trends and the fundamental algorithms, the material presented here offers a useful code base.

In a more hands-on computational linguistics class, making the `Kim` or `Kornai` systems work with the `Watson` toolkit would be a challenging but rewarding project. To bring advanced computational linguistics to the masses would require an even more ambitious effort, such as building a copylefted rule compiler (which could be used e.g. to model the `Uniparse` system, though not necessarily in the historically faithful manner presented here) or recreating the AT&T/Bell Labs framework (see `http://www.research.att.com/sw/tools/fsm`).

As usual, everything on the cd-rom is suplied "as is" and carries no warranty, express or implied, as to merchantibility, fitness for a particular purpose, title, or anything else. Users shall indemnify and hold the publisher, the original authors, and their employers harmless from and against any loss, claim, or demand arising out of the use of the software. Users are expected to respect the copyright notices included with the material, and if they produce readily `make`-able packages these

should be placed under GNU General Public License, Artistic License, or other forms of copyright protection designed to keep the material accessible for further hacking.

To understand some of the main trends in finite state NLP it is worth looking back at the origins of the field. Though neither Mealy (1955) nor Kleene (1956) had NL applications in mind, finite state methods were applied in this domain as early as 1958. The rediscovery of this work (see **Joshi**'s paper, **Karttunen**'s comments, and `/mnt/cdrom/Uniparse`) has been one of the pleasant surprises of the ECAI workshop. In the early sixties, however, finite state models were soon submerged in a flood of transformational models. At that time neither careful attendance to linguistic detail nor husbanding of computational resources held much appeal, and the excitement generated by the breathtaking pace of development from Syntactic Structures to Aspects and the Standard Model, the Extended Standard Model, and the Revised Extended Standard Model kept most computational linguists too busy to think through the implications.

It is hard to speculate about such matters, but it is quite conceivable that the finite state approach to NLP would have lost all credibility, were it not for the extraordinary impact of Thompson (1968) and the `grep` family of unix tools. While theoretical linguists accepted the arguments put forth in Miller and Chomsky (1963) at face value, from the seventies it became part of the received computer science wisdom that if you want to do something with text you need to build finite automata. By making his implementation of `regexp(3)` freely redistributable, Spencer (1986) transmitted this wisdom to the free software movement, and subsequent works including GNU `flex` and `agrep` (Wu and Manber 1992) have spread to many corners of computer science from compilers to protocols. In this volume this trend is represented by the FIRE Lite toolkit described by **Watson** and made available under `/mnt/cdrom/Watson`. The customizable software presented here finally brings computations involving automata with hundreds of thousands or even millions of states outside the confines of highly proprietary development environments. As automata grow in size, it is becoming increasingly important to develop tools for their testing and debugging, and the work described in **Vilares** *et al.* is a good first step in this direction.

Given the dominant position of finite state technologies in topic search, in retrospect it is hard to understand why mainstream syntactic theory continued to shun finite state methods throughout the seventies and eighties, but in fact these methods reappeared on the scene through a back door left open by the context sensitive rule systems of phonology. Only two years after the seminal Sound Pattern of English (Chomsky and Halle 1968), Johnson (1970) demonstrated that the context sensitive machinery of SPE can be replaced by a much simpler one, based on finite state transducers (FSTs), and independently the same conclusion was reached by Kaplan and Kay, whose work remained an underground classic until it was finally published in (Kaplan and Kay1994). Eventually, computational

linguists interested in describing the wealth of detail present in the phonology and morphology of agglutinative languages became frustrated with the problem of context sensitive parsing, and the practical solution offered by Koskenniemi (1983), propelled both by the Xerox rule compiler (Dalrymple *et al.* 1987) and by Antworth's (1990) PC implementation, became the dominant computational model in the field. To this day, the dominant finite state paradigm is the Xerox regular expression calculus, as exemplified by the **Tateno** *et al.* paper on the Japanese lexical transducer. The paper by **Kim and Jang** and the accompanying software under `/mnt/cdrom/Kim` presents a somewhat different use of finite state automata in Korean morphology.

Finite state syntax, though advocated by a minority throughout the eighties (**?**; Kornai 1985), did not really come in from the cold until the nineties. The present volume offers some prime examples of this work in the papers by **Chanod and Tapanainen, Grefenstette, Kornai,** and **Roche**, who employ finite state methods to describe phenomena, such as light verbs, which were in the tradition of Chomsky (1970) treated as core cases of transformational grammar. The paper by **Vilar** *et al.* describes finite state methods of machine translation, and **Ejerhed**'s paper pushes the envelope even further, by offering a finite state model of key discourse phenomena. Another important way in which mainstream syntax is impacted by finite state techniques can be called "finite state to the rescue" – the paper by **Schulz and Mikołajewski** describes how constraint-based grammars can be speeded up by finite state methods, and the paper by **Srinivas** shows how corpus-based acquisition of LTAGs is facilitated by finite state techniques.

Perhaps the clearest sign that finite state approaches became part of the mainstream is that they are now subject to the same trends as the rest of computational linguistics. In particular, we see an increased interplay between the statistical and rule-based paradigms in this domain. In some part this is due to the finite state nature of much statistical work (in fact the founding paper of the field, Markov (1913) can be seen in retrospect as a finitary model) but in greater part it is due to an increased awareness on the part of grammar writers that certain aspects of the system, most notably the relationship between the spoken, written, or signed signal and the underlying psychological units, resist characterization in non-statistical terms. An important step in bringing rule-based and statistical work closer is the framework of weighted transducers developed at AT&T/Bell Labs, represented in this volume by the **Tzoukerman and Radev** paper.

While it is certainly true that the mathematical theory of (weighted) regular sets and relations is mature, the same can not be said of the algorithmic aspects of the subject. As the size of the machines grows, we can discern two complementary trends: on the one hand, the search for more efficient algorithms continues, and on the other, techniques leveraging the already remarkable efficiency and scalability of finite state techniques begin to appear. Since some excellent descriptions of the main algorithms and model building techniques are already available (see in particular (Watson 1995), (Roche and Schabes 1997)), in this volume we concen-

trate on the second trend, extending the scope of finite state machinery. The more complex formal systems discussed by **Csuhaj-Varjú, Nederhof and Bertsch,** and **Ristad** are likely to provide a fertile ground for further experimentation with extended finite state models of language.

Acknowledgements

This volume could not have come into being without the work of the other workshop organizers, Eva Ejerhed (chair), Frederic Jelinek, and Lauri Karttunen. In addition to them, special thanks are due to the other referees, Salah Ait-Mokhtar, Vagelatos Aristides, Erzsébet Csuhaj-Varjú, Aravind Joshi, Fred Karlsson, Kimmo Koskenniemi, Doug Merritt, Mehryar Mohri, Emmanuel Roche, Richard Sproat, and those who wished to remain anonymous. The cd-rom benefited a great deal from the work of Philip Hopely.

2 A parser from antiquity: an early application of finite state transducers to natural language parsing

ARAVIND K. JOSHI AND PHILIP HOPELY

Abstract

This paper describes the key aspects of a parser developed at the University of Pennsylvania from 1958 to 1959. The parser is essentially a cascade of finite state transducers. To the best of our knowledge, this is the first application of finite state transducers to parsing. This parser was recently faithfully reconstructed from the original documentation. Many aspects of this program have a close relationship to some of the recent work on finite state transducers.

1 Introduction

A parsing program was designed and implemented at the University of Pennsylvania during the period from June 1958 to July 1959. This program was part of the Transformations and Discourse Analysis Project (TDAP) directed by Zellig S. Harris. The techniques used in this program, besides being influenced by the particular linguistic theory, arose out of the need to deal with the extremely limited computational resources available at that time. The program was essentially a cascade of finite state transducers (FSTs). To the best of our knowledge, this is the first application of FSTs to parsing. The program consisted of the following phases:

1. Dictionary look-up.
2. Replacement of some 'grammatical idioms' by a single part of speech.
3. Rule based part of speech disambiguation.
4. A right to left FST composed with a left to right FST for computing 'simple noun phrases.'
5. A left to right FST for computing 'simple adjuncts' such as prepositional phrases and adverbial phrases.
6. A left to right FST for computing simple verb clusters.
7. A left to right 'FST' for computing clauses.

In Section 2 we will describe the different phases of the parser in some detail and also briefly discuss several aspects of the parser that have a close relationship to some of the recent work on finite state transducers. An illustrative example is provided in Section 3 showing the output of each phase. This is followed by a brief description of the reconstruction and evaluation of the parser in Section 4.