A Parser from Antiquity: An Early Application of Finite State Transducers to Natural Language Parsing

Aravind K. Joshi
Department of Computer and Information Science
Room 555 Moore School
University of Pennsylvania
Philadelphia, Pa 19104 USA
joshi@linc.cis.upenn.edu

A parsing program was designed and implemented at the University of Pennsylvania during the period 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 (fst). To the best of our knowledge, this is the first application of fst's 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 parts- 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 Phase 1, each word is assigned one or more parts-of-speech (POS). If a word is assigned more than one POS, then sometimes they are ranked, the less frequent POS first and then the next. Thus for example, for 'show' N is ranked before V and for 'book' V is ranked before N. There were about 14 verb subcategorizations for verbs. Since the Prepositional Phrases (PPs) were marked with the specific prepositions, there were effectively over 50 subcategorizations. The parser did not handle unknown words.

In Phase 2, A 'grammatical idioms' such as 'of course' is replaced by a single POS for adverb, 'per cent' by POS for noun etc. For each 'grammatical idiom' one word is marked as an index in the dictionary together with the local environment (words to the left and to the right of the index word), also specified in the dictionary. Phase 2 is a simple case of finite transduction.

In Phase 3, rule based disambiguation techniques are used for POS disambiguation. There were about 14 tests, N-eliminating tests, V-eliminating tests, etc. If the POS for a word are ordered, for example, for 'show' N before V, then the N-eliminating tests are applied first. If they fail then the V-eliminating tests area applied. If these fail also then the ambiguity remains. Most tests look for bounded contexts to the left and to the right; thus these are finite transductions. However some tests use contexts specifiable by simple regular expressions and thus they are finite state transductions. The ordered set of tests are cycled until no further disambiguations can be made.

The strings (phrases) computed in Phases 4, 5, and 6 above are called first-order strings as they do not involve proper nesting. The strings (clauses) computed in Phase 7 are called second order strings as they may involve proper nestings. The computation in Phase 7 is strictly speaking not a finite state computation.

The fst's were made 'effectively' deterministic by (1) choosing the direction of the scan (left-to-right or right-to-left) and adopting the longest path strategy, (2) cascading right-to-left and left-to-right transductions, and (3) using the delimiting characters to allow for some minimal nondeterminism. These aspects of the program have a close relationship to some of the recent work on fst's such as subsequential machines [7,9], decomposition of an fst into a two sequential fst's [5,8] and the work on 'directed replacement' [6]. The parsing style itself has resemblance to Abney's chunking parser [4].

The overall objective of the program was to prepare the text for tasks such as abstracting. However, the 1958-59 program only did the parsing. Besides parsing a large number of test sentences, the program processed about 25 sentences from a journal paper in biochemistry. Although the fst's compute more structure, the final output shows relatively flat structures. Adjuncts are never explicitly attached. Here is an example:

(1) We have found that subsequent addition of the second inducer of either system after allowing single induction to proceed for 15 minutes also results in increased reproduction of both enzymes

There are no grammatical idioms in this example. In Phase

Proceedings of the ECAI 96 Workshop
Extended Finite State Models of Language
Edited by A. Kornai.

3, 'results' (N/V) is resolved to V. After the first three phases, the first right-to-left fst identifies the following simple NPs in this example, enclosed in [...].

(2) [We] have found that [subsequent addition] of [the second inducer] of [either system] after allowing [single induction] to proceed for [15 minutes] also results in [increased reproduction] of [both enzymes]

The next left-to-right fst does not identify any new simple NPs in this example. It would have found NPs such as [the rich], which are identified in the left-to-right scan.

The next left-to-right fst identifies the following simple adjuncts in this example, enclosed in (...).

(3) [We] have found that [subsequent addition](of [the second inducer])(of [either system]) after allowing [single induction] to proceed (for [15 minutes]) (also) results (in [increased reproduction]) (of [both enzymes])

The next left-to-right fst identifies the following simple verb clusters in this example, enclosed in { ... }.

(4) [We]{ have found } that [subsequent addition](of [the second inducer])(of [either system]) after { allowing } [single induction] to proceed (for [15 minutes]) (also) {results} (in [increased reproduction]) (of [both enzymes])

The final left-to-right scan identifies the clauses, enclosed in < ... >. The main clauses is not enclosed in any brackets. + indicates end of a complement. Thus the final output is as follows.

(5) [We] {have found} that [subsequent addition](of [the second inducer])(of [either system]) < after {allowing} [single induction] to proceed + > (for [15 minutes]) (also) {results} (in [increased reproduction]) + > + (of [both enzymes])

In each one of these phases the longest path criterion is used. This results in longest simple NPs, simple adjuncts, simple verb clusters and clauses. While looking for verb complements the longest complement is preferred.

Recently this program was faithfully reconstructed, a collaborative effort with Phil Hopely, from the original documentation, which fortunately exists [3] and it is in sufficient detail to make the reconstruction possible. The reconstructed parser (now called Uniparse) has also been tested on about 50 sentences from each of the three corpora–Wall Street Journal, IBM computer manuals, and ATIS.

We will discuss this program focusing on the fst aspects, relating them, where appropriate, to some of the recent work on fst. We will also briefly describe the performance of Uniparse on the small set of sentences of the three corpora mentioned above.

Historical note:

The original program was implemented (in the assembly language) on Univac 1, a single user machine. The machine had acoustic delay line (mercury delay line) memory of 1000 words. Each word was 12 characters/digits, each character/digit was 6 bits.

Lila Gleitman, Aravind Joshi, Bruria Kauffman, and Naomi Sager and a little later, Carol Chomsky were involved in the development and implementation of this program. A brief description of the program appears in [2] and a somewhat generalized description of the grammar appears in [1]. This program is the precursor of the string grammar program of Naomi Sager at NYU, leading up to the current parsers of Ralph Grishman (NYU) and Lynette Hirschman (formerly at UNISYS, now at Mitre Corporation). Carol Chomsky took the program to MIT and it was used in the question-answer program of Green, BASEBALL (1961). At Penn, it led to a program for transformational analysis (kernels and transformations) (1963) and, in many ways, influenced the formal work on string adjunction (1972) and later tree-adjunction (1975).

REFERENCES (for Uniparse)

[1] Zellig S. Harris, String Analysis of Sentence Structure, Mouton & Co. The Hague, (1962).

[2] Aravind K. Joshi, 'Computation of Syntactic Structure', in *Advances in Documentation and Library Science*, vol III, part 2 , Interscience Publishers, (1961).

[3] Transformations and Discourse Analysis Project (TDAP) Reports, University of Pennsylvania, Reports #15 through #19, 1959-60. Available in the Library of the National Institute of Science and Technology (NIST) (formerly known as the National Bureau of Standards (NBS)), Bethesda, MD.

OTHER REFERENCES

[4] Steven Abney, 'Parsing by chunks', in *Principle-based Parsing* (eds. Robert Berwick and Steven Abney and Carol Tenny), Kluwer Academic Publishers, (1991).

[5] Calvin C. Elgot and J.E. Mezzi, 'On relations defined by generalized finite automata', *IBM Journal of Research and Development*, 9, (1965).

[6] Lauri Karttunen, 'Directed replacement', in *Proceedings of the 34th Annual Meeting of ACL*, Santa Cruz, CA, (1996).

[7] Mehryar Mohri, 'Finite-state transducers in language and speech processing', *Computational Linguistics*, 20:1 1-34, (1996).

[8] Emmannuel Roche, 'Two parsing methods by means of finite state transducers', in *Proceedings of the 16th International Conference on Computational Linguistics* (COLING'94), Kyoto, Japan, (1994).

[9] Marcel P. Schutzenberger, 'Sur une variante des fonctions séquentielles, *Theoretical Computer Science*, (1977).