Comparative Study of GLR Parser with Finite-state Predictors and Chart-based Semantic Parsers

RACHITA SHARMA AND SANJAY KUMAR DUBEY
Greater Noida Institute of Technology, Greater Noida (India)
E-mail: anulibra09@gmail.com

Sanjay Kumar Dubey, Amity School of Engineering and Technology, Amity University, Sec-125, Noida, (U.P.), India, E-mail: skdubey1@amity.edu

Abstract: The natural language processing component of a speech understanding system is commonly a robust, semantic parser, implemented as either a chart-based transition network, or as a generalized left right (GLR) parser. In contrast, we are developing a robust, semantic parser that is a single, predictive finite-state machine. Our approach is motivated by our belief that such a finite-state parser can ultimately provide an efficient vehicle for tightly integrating higher-level linguistic knowledge into speech recognition. We will describe the development of this parser, with an example of its use, and a description of how it compares to both finite-state predictors and chart-based semantic parsers, while combining the elements of both.

Keywords: GLR, PDA, LR Parser, Compiler

1. INTRODUCTION: GLR Parsing
The GLR* parsing algorithm developed in this thesis is based on The Generalized LR (GLR) parsing algorithm that was developed by Tomita.

1.1 Principles of LR Parsing
The GLR parsing algorithm evolved out of the LR parsing techniques that were originally developed for parsing programming languages in the late 1960s and early 1970s. LR parsers parse the input bottom-up, scanning it from left to right, and producing a rightmost derivation. They are deterministic and extremely efficient, being driven by a table of parsing actions that is pre-compiled from the grammar.

The common core of all of the LR parser variants is a basic Shift-Reduce parser, which is fundamentally no more than a finite-state pushdown automaton (PDA). The parser scans a given input left to right, word by word. At each step, the LR parser may either shift the next input word on to the stack, reduce the current stack according to a grammar rule, accept the input, or reject it. An action table that is pre-compiled from the grammar guides the LR parser when parsing an input. The action table specifies the next action that the parser must take, as a function of its current state and the next word of the input.

1.2 The GLR Parsing Algorithm
Tomita’s Generalized LR parsing algorithm extended the original LR parsing algorithm to the case of non-LR languages, where the parsing tables contain entries with multiple parsing actions. The
algorithm deterministically simulates the non-determinism introduced by the conflicting actions in the parsing table by efficiently pursuing in a pseudo-parallel fashion all possible actions. The primary tool for performing this simulation efficiently is the Graph Structured Stack (GSS). Two additional techniques, local ambiguity packing and shared parse forests, are used in order to efficiently represent the various parse trees of ambiguous sentences when they are parsed.

1.3 The Unification-based GLR Parsing System

The Generalized LR Parser/Compiler is a unification based practical natural language system that was designed around the GLR parsing algorithm at the Center for Machine Translation at Carnegie Mellon University. The system supports grammatical specification in an LFG framework, that consists of context-free grammar rules augmented with feature bundles that are associated with the non-terminals of the rules. Feature structure computation is, for the most part, specified and implemented via unification operations. This allows the grammar to constrain the applicability of context-free rules. A reduction by a context-free rule succeeds only if the associated feature structure unification is successful as well. The Generalized LR Parser/Compiler is implemented in Common Lisp, and has been used as the analysis component of several different projects at the Center for Machine Translation at CMU in the course of the last several years.

2. THE GLR* PARSING ALGORITHM

Chapter 3 of the thesis describes the GLR* parsing algorithm, analyzes its computational complexity and evaluates its practical performance. It also describes the search control heuristics that were developed to ensure that the algorithm performs within feasible time and space bounds, and evaluates the effectiveness of these heuristics.

The GLR* parsing algorithm was designed to be robust to two particular types of extra grammaticality: noise in the input, and limited grammar coverage. It attempts to overcome these forms of extra-grammaticality by ignoring the unparsable words and fragments and conducting a search for the maximal subset of the original input that is covered by the grammar.

2.1 The Unrestricted GLR* Parsing Algorithm

GLR* accommodates skipping words of the input string by allowing shift operations to be performed from inactive state nodes in the GSS. Shifting an input symbol from an inactive state is equivalent to skipping the words of the input that were encountered after the parser reached the inactive state and prior to the current word that is being shifted. Since the parser is LR(0), previous reduce operations remain valid even when words further along in the input are skipped, since there ductions do not depend on any look ahead. Due to the word skipping behavior of the GLR* parser, local ambiguities occur on a much more frequent basis than before. In many cases, a portion of the input sentence may be reduced to a non-terminal symbol in many different ways, when considering different subsets of the input that may be skipped. Since the ultimate goal of the GLR* parser is to produce maximal (or close to maximal) parses, the process of local ambiguity packing can be used to discard partial parses that are not likely to lead to the desired maximal parse. Local ambiguities that differ in their coverage of an input segment can be compared, and when the word coverage of one strictly subsumes that of another, the subsumed ambiguity can be discarded.
2.2 Complexity Analysis and Performance Evaluation

The complexity analysis of the unrestricted GLR* algorithm proves that the asymptotic time complexity of the algorithm is the longest grammar rule. This complexity bound is similar to that of the original GLR parsing algorithm. This result may seem surprising, but we provide several explanations for it. With respect to the grammar size, GLR has a non-polynomially-bound runtime, and once again, GLR* has a similar bound. These complexity results thus do not provide much insight into the actual differences in runtime behavior between GLR and GLR* on common practical grammars. We therefore conducted experiments to evaluate the practical performance of the unrestricted GLR* algorithm. Time and space performance of the unrestricted GLR* parser was evaluated on a benchmark setting from the JANUS speech-to-speech translation project. We used a version of the English analysis grammar for the scheduling domain. The benchmark consists of 552 English sentences, with sentence length ranging from 2 to 15 words, and an average length of about 6 words per sentence. The evaluation results show that the time and space behavior of the unrestricted GLR* parser diverges very rapidly from that of GLR, and becomes infeasible for even sentences of medium length. Whereas GLR time and space requirements increase almost linearly as a function of the sentence length, the performance of unrestricted GLR* appears to be similar to that predicted by the worst case complexity analysis. This suggests that effective search heuristics are required to ensure that the algorithm performs within feasible time and space bounds.

2.3. The Search Control Heuristics

Since the purpose of GLR* is to find only maximal (or close to maximal) parsable subsets of the input, we can drastically reduce the amount of search by limiting the amount of word skipping that is considered by the parser. We developed and experimented with two heuristic search techniques. With the k-word Skip Limit Heuristic, the parser is restricted to skip no more than Consecutive input words at any point in the parsing process. With the Beam Search Heuristic, the parser is restricted to pursue a “beam” of a fixed size of parsing actions, which are selected according to a criterion that locally minimizes the number of words skipped. There exists a direct trade off between the amount of search (and word skipping) that the parser is allowed to pursue and the time and space performance of the parser itself. The goal is to determine the smallest possible setting of the control parameters that allows the parser to find the desired parses in an overwhelming majority of cases. We conducted empirical evaluations of parser behavior with different settings of the search control parameters, using the same benchmark setting on which the unrestricted GLR* parser was evaluated. The results show that time and space requirements of GLR* increase as the control parameters are set to higher values. This increase however is gradual, and in the case of the beam parameter, even with considerably high settings, parser performance remains in the feasible range, and does not approach the time and space requirements experienced when running the unrestricted version of GLR*. A comparison of the performance figures of the two heuristic control mechanisms revealed a clear advantage of using the beam search, versus the simpler skip word limit heuristic.

3. CONCLUSIONS

We have outlined our method for constructing a predictive, robust finite-state parser, and contrasted it with both FSAs and the PHOENIX system, a robust chart-based semantic parser. We have also demonstrated the use of this parser within a small prototype system, built within the CSLU toolkit’s RAD environment.
Although our work on PROFER is still in its early stages, we have shown that such an approach is viable, both as a stand-alone semantic parser, and as a stand-alone finite-state predictor. In the future we hope to show that such an approach will also be viable for tightly integrating higher-level linguistic constraints into the speech recognition process.

References


