Practical Ambiguity Detection for Context-Free Grammars
Research Abstract

H. J. S. Basten

Centrum Wiskunde & Informatica

Abstract. The use of unconstrained context-free grammars for generalized parsing techniques has several advantages over traditional grammar classes, but comes with the danger of undiscovered ambiguities. The ambiguity problem for these grammars is undecidable in the general case, but this does not have to be a problem in practice. Our goal is to find ambiguity detection techniques that have sufficient precision and performance to make them suitable for practical use on realistic grammars. We give a short overview of related work, and propose new directions for improvement.

1 Problem Description and Motivation

Generalized parsing techniques allow the use of the entire class of context-free grammars (CFGs) for the specification of the syntax of programming languages. This has several advantages. First, it allows for modular syntax definitions, which simplifies grammar development and enables reuse. Second, it grants total freedom in structuring a grammar to best fit its intended use. Grammars do not have to be squeezed into LL, LALR or LR(k) form for instance.

Unfortunately, using unconstrained context-free grammars comes with the danger of ambiguities. A grammar is ambiguous if one or more sentences in its language have multiple parse trees. The semantics of a sentence is usually based upon the structure of its parse tree, so an ambiguous sentence can have multiple meanings. This often indicates a *grammar bug* which should be avoided. However, in some cases a grammar is intended to contain some degree of ambiguity. For instance in reverse engineering, where certain legacy languages can only be disambiguated with type checking after parsing. In both cases it is important to know the sources of ambiguity in the developed grammar, so they can be resolved or verified.

Unfortunately, detecting the (un)ambiguity of a grammar is undecidable in the general case [7, 10, 9]. However, this does not necessarily have to be a problem in practice. Several Ambiguity Detection Methods (ADMs) exist that approach the problem from different angles, all with their own strengths and weaknesses. Because of the undecidability of the problem there is a general tradeoff between precision and performance/termination. The challenge for all ADMs is to give the most precise and understandable answer in the time available. The current state of the art is not yet sufficiently advanced to be practical on realistic grammars, especially the larger ones.

2 Brief Overview of Related Work

Existing ADMs can roughly be divided into two categories: *exhaustive* methods and *approximative* ones. Methods in the first category exhaustively search the language of a grammar for ambiguous sentences. This so called *sentence generation* is applied by [11, 8, 13, 1]. These methods are 100% accurate, but a problem is that they never terminate if the grammar's language is of infinite size, which usually is the case. They do produce the most precise and useful ambiguity reports, namely ambiguous sentences and their parse trees.

Approximative methods sacrifice accuracy to be able to always finish in finite time. They search an approximation of the grammar for possible ambiguity. The methods described in [12, 6] both apply conservative approximation to never miss ambiguities. The downside of this is that when they do find ambiguities, it is hard to verify whether or not these are false positives.

In [2] we compared the practical usability of several ADMs on a set of grammars for real world programming languages. It turned out that the exhaustive sentence generator AMBER [13] was the most practical due to its exact reports and reasonable performance. However, it was still unsatisfactory to find realistic ambiguities in longer sentences. The approximative Noncanonical Unambiguity test [12] had a reasonably high accuracy, but it is only able to assess the ambiguity of a grammar as a whole. Its reports might point out sources of individual ambiguities, but these can be hard to understand.

3 Proposed Solution

The aim of this research is to increase the precision and performance of ambiguity detection to a practical level. More specifically, the idea is to increase the performance of exhaustive searching by reducing the search space using approximative techniques. For instance by identifying *harmless production rules* in a grammar. These are rules that are certainly not used in the derivation of any ambiguous string. Since these rules do not contribute to the ambiguity of the grammar they can be removed before exhaustive searching is applied, which reduces the number of sentences to generate.

In [3] we describe a first exploration in this direction. We propose an extension to the approximative Noncanonical Unambiguity test that enables it to identify harmless production rules. We implemented this filtering technique into a tool [4], which we applied on a series of real world programming language grammars in [5]. It is shown that the performance of the sentence generators AMBER and CfgAnalyzer is indeed improved by several orders of magnitude, with only a small filtering overhead.

Further research will be focussed on finding more detailed detection techniques that can identify harmless rules with more precision, as well as faster sentence generation methods. For instance by exploring opportunities for paralellisation. Furthermore, we like to extend existing techniques to include commonly used disambiguation constructs, like priorities and associativities, longest match, keyword reservation, etc.

4 Research Method

New techniques will be validated both theoretically and experimentally. Through formal specification they will be proved correct. Then, to test a technique's suitability for practical use, a prototype implementation will be tested on a series of realistic benchmark grammars. For instance grammars for real world programming languages, that will be seeded with ambiguities if needed.

5 Conclusion

This abstract proposes research into ambiguity detection for context-free grammars, to make it suitable for practical use. More specifically, it aims at combining approximative searching with exhaustive searching, to be able to find real ambiguous sentences in shorter time. First explorations in this direction show promising results.

References

1. Axelsson, R., Heljanko, K., Lange, M.: Analyzing context-free grammars using an incremental SAT solver. In: Proceedings of the 35th International Colloquium on Automata, Languages, and Programming (ICALP 2008). LNCS, vol. 5126 (2008)
2. Basten, H.J.S.: The usability of ambiguity detection methods for context-free grammars. In: Johnstone, A., Vinju, J. (eds.) Proceedings of the Eigth Workshop on Language Descriptions, Tools and Applications (LDTA 2008). ENTCS, vol. 238 (2009)
3. Basten, H.J.S.: Tracking down the origins of ambiguity in context-free grammars. In: Proceedings of the Seventh International Colloquium on Theoretical Aspects of Computing (ICTAC 2010). Springer (2010), To appear, see `www.cwi.nl/~basten` for a preliminary version.
4. Basten, H.J.S., van der Storm, T.: AmbiDexter: Practical ambiguity detection, tool demonstration. In: Proceedings of the Tenth IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM 2010). IEEE (2010), To appear, see `www.cwi.nl/~basten` for a preliminary version.
5. Basten, H.J.S., Vinju, J.J.: Faster ambiguity detection by grammar filtering. In: Proceedings of the Tenth Workshop on Language Descriptions, Tools and Applications (LDTA 2010). ACM (2010), To appear, see `www.cwi.nl/~basten` for a preliminary version.
6. Brabrand, C., Giegerich, R., Møller, A.: Analyzing ambiguity of context-free grammars. Science of Computer Programming 75(3), 176–191 (2010)
7. Cantor, D.G.: On the ambiguity problem of Backus systems. Journal of the ACM 9(4), 477–479 (1962)

8. Cheung, B.S.N., Uzgalis, R.C.: Ambiguity in context-free grammars. In: Proceedings of the 1995 ACM Symposium on Applied Computing (SAC 1995). pp. 272–276. ACM Press, New York, NY, USA (1995), `http://doi.acm.org/10.1145/315891.315991`
9. Chomsky, N., Schützenberger, M.: The algebraic theory of context-free languages. In: Braffort, P. (ed.) Computer Programming and Formal Systems, pp. 118–161. North-Holland, Amsterdam (1963)
10. Floyd, R.W.: On ambiguity in phrase structure languages. Communications of the ACM 5(10), 526–534 (1962)
11. Gorn, S.: Detection of generative ambiguities in context-free mechanical languages. J. ACM 10(2), 196–208 (1963), `http://doi.acm.org/10.1145/321160.321168`
12. Schmitz, S.: Conservative ambiguity detection in context-free grammars. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP'07: 34th International Colloquium on Automata, Languages and Programming. LNCS, vol. 4596 (2007)
13. Schröer, F.W.: AMBER, an ambiguity checker for context-free grammars. Tech. rep., compilertools.net (2001), see `http://accent.compilertools.net/Amber.html`