Restricting Grammars with Tree Automata

VYPa 2023 – presentation abstract

Authors: Ján Maťufka (`xmatuf00`), Michal Pyšík (`xpysik00`)

One of the drawbacks of using context-free grammars (CFGs) in compiler design is their ambiguity. This often leads compiler (and programming language) designers to refactor grammars to eliminate unintended ambiguity. Another limiting factor is that context-free languages (and their models) are not closed under intersection, complementation, and difference. If the opposite were true, then it would be easier to construct intended constraints encoding unwanted parsing possibilities and intersecting them with the initial CFG.

Although such an approach is impossible in CFGs, it is feasible in tree automata (TAs). Instead of a set of strings, TAs represent a set of parse trees. Ambiguous CFGs can represent one string with multiple parse trees (obtained from the same grammar), whereas a tree automaton will generate only a single parse tree from such a string, given a certain set of rules (or none at all if the string cannot be generated from the original grammar).

The proposed approach shows a way to convert every CFG into a TA, such that the set of trees from the TA language accepts every possible parse tree (or derivation tree) representing derivation steps on words from the language generated by the CFG. By converting initial (ambiguous) CFG into a TA and by creating TAs with the desired parse tree restrictions, one can obtain a TA that will solve targeted ambiguities and thus create a more desirable (less ambiguous) grammar.

Such an approach could be used to resolve precedence, associativity, dangling `else`, and other language-specific simplifications. It is shown that imposing restrictions on CFGs using tree automata can produce correct grammars with almost the same parsing time (i.e., the parser using such a "restricted" grammar will perform well). It is also shown that intersecting the "restriction" TA with the "grammar" TA will not introduce new ambiguities. In the worst case, it will duplicate already existing ones.

Sources

[1] Adams, M. D. and Might, M. Restricting grammars with tree automata. *Proceedings of the ACM on Programming Languages*. Association for Computing Machinery (ACM). october 2017, vol. 1, OOPSLA, p. 1–25. DOI: 10.1145/3133906. Available at: `https://dl.acm.org/doi/pdf/10.1145/3133906`.

[2] Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F. et al. *Tree Automata Techniques and Applications*. 2007. Release October, 12th 2007. Available at: `http://www.grappa.univ-lille3.fr/tata`.