Certified Parsing

Background

Parsers transform plain text into some abstract structure that can be analyzed by computers for further processing. One might think that parsers have been studied to death, and after *yacc* and *lex* no new results can be obtained in this area. However recent developments and novel approaches make it increasingly clear, that this is not true anymore [8]. And there is a practical need for new results: for example the future HTML 5 Standard abandons a well-defined grammar specification, in favour of a bespoke parser given as pseudo code.

This work targets parsers from a certification point of view. Increasingly, parsers are part of certified compilers, like *CompCert* [7], which are guaranteed to be bug-free. Such certified compilers are crucial in areas where software just cannot fail. However, so far the parsers of these compilers have been left out of the certification. This is because parsing algorithms are often ad hoc and their semantics is not clearly specified. This means, unfortunately, parsers can harbour errors that potentially invalidate the whole certification and correctness of the compiler. In this project, we aim to change this situation with the help of the theorem prover Isabelle/HOL.

Only in the last few years, theorem provers have become powerful enough for establishing the correctness of some standard lexing and parsing algorithms. For this, the algorithms still need to be formulated in way so that it is easy to reason about them. In our earlier work about lexing and regular languages, we showed that this precludes well-known algorithms based automata [12]. However we showed also that regular languages can be formulated and reasoned about entirely in terms regular expressions, which can be easily represented in theorem provers. This work uses the device of derivatives of regular expressions. We like to extend this device to parsers and grammars. The aim is to come up with elegant and practical useful parsing algorithms whose correctness can be certified in Isabelle/HOL.

Proposed Work

A recent development in the field of parsing are Parsing Expression Grammars (PEGs) [5], which are an extension of the well-known Context Free Grammars (CFGs). This extension introduces new regular operators, such as negation and conjunction, on the right-hand side of grammar rules, as well as priority orderings for rules. With these extensions, PEG parsing becomes much more powerful and more useful in practise. For example disambiguation, formerly expressed by semantic filters, can now be expressed directly using grammar rules.

However, there is a serious limitation of PEGs, which affects potential applications: grammars involving left recursion are not allowed [4]. Although one PEG parsing algorithm has already been proposed that can deal with left recursion [11], there is no correctness proof for it, not even one with “pencil and paper”. One aim of this research is to solve this sorry state-of-affairs by either certifying this algorithm or inventing a new one. For this we will first devise a fixed-point semantics of PEGs, against which we can certify a parser. For this semantics we take as starting point the paper [5], which does not treat left-recursion, but gives an operational semantics for PEG parsing. There are also good indications that we can adapt work on Boolean Grammars [9], which are similar to PEGs, and for which the paper [6] gives a fixed-point semantics to negation operators, but not to the Kleene star.

For the parsing algorithm, we might be able to draw inspiration from parsers based on the classic Cocke-Younger-Kasami (CYK) algorithms [6] and Early [1, 3] parsers. The defect of CYK algorithms, however, is that the grammar specifications given by the user need to be transformed into a normal form. This transformation may potentially lead to rule explosion and hence inefficient parsing. We will investigate whether this transformation can be avoided. The problem with Early-style parsers is that they need to be extended to PEG parsing in order to be helpful for us.

Finally, we want to investigate whether derivatives of regular expressions [2, 8, 10] can be generalised to PEG parsing. In earlier work, we showed that lexing based on derivatives gives rise to very elegant regular expression matchers that can be certified in a theorem prover with ease. We will study whether the idea of taking a derivative of a regular expression can be extended to rules in grammars. The problem that needs to be overcome is again how to deal with possible left-recursion in grammar rules.

References

[1] J. Aycock and R. N. Horspool. Practical Earley Parsing. *The Computer Journal*, 45(6):620–630, 2002.

[2] J. A. Brzozowski. Derivatives of Regular Expressions. *Journal of the ACM*, 11(4):481–494, 1964.

[3] J. Earley. An Efficient Context-Free Parsing Algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[4] B. Ford. Packrat Parsing: : Simple, Powerful, Lazy, Linear Time, (Functional Pearl). In *Proc. of the 7th ACM International Conference on Functional Programming (ICFP)*, pages 36–47, 2002.

[5] B. Ford. Parsing Expression Grammars: A Recognition-Based Syntactic Foundation. In *Proc. of the 31st ACM Symposium on Principles of Programming Languages (POPL)*, pages 111–122, 2004.

[6] V. Kountouriotis, C. Nomikos, and P. Rondogiannis. Well-founded Semantics for Boolean Grammars. *Information and Computation*, 207(9):945–967, 2009.

[7] X. Leroy. Formal Verification of a Realistic Compiler. *Communications of the ACM*, 52(7):107–115, 2009.

[8] M. Might and D. Darais. Yacc is Dead. To appear in *Proc. of the 16th ACM International Conference on Functional Programming (ICFP)*, 2011.

[9] A. Okhotin. Boolean Grammars. *Information and Computation*, 194(1):19–48, 2004.

[10] S. Owens, J. Reppy, and A. Turon. Regular-Expression Derivatives Re-Examined. *Journal of Functional Programming*, 19(2):173–190, 2009.

[11] A. Warth, J. R. Douglass, and T. D. Millstein. Packrat Parsers Can Support Left Recursion. In *Proc. of the ACM Symposium on Partial Evaluation and Semantics-based Program Manipulation (PEPM)*, pages 103–110, 2008.

[12] C. Wu, X. Zhang, and C. Urban. A Formalisation of the Myhill-Nerode Theorem based on Regular Expressions (Proof Pearl). In *Proc. of the 2nd International Conference on Interactive Theorem Proving*, volume 6898 of *LNCS*, pages 341–356, 2011.