LALR(1) parsing for the 21st century

Ted Kaminski & Eric Van Wyk
University of Minnesota
tedisnki,evw@cs.umn.edu

LALR(1) parsing has earned an undeservedly bad reputation for being brittle and difficult to use, many citing difficulties in massaging a grammar to conform to the restrictions of the LALR(1) class and arcane error messages from parser generators when the grammar is not in this class. We have found, however, that many of the stated reasons for using other parsing technologies are, in fact, not inherently problems with deterministic LR(1) or LALR(1) parsing, but with the particular tools and ways of using these kind of parsers.

The most-used LALR parser generators continue to disavow anything to do with scanning – an old design choice that no longer seems to be the right one. Copper, Schwerdfeger's LALR(1) parser and context-aware scanner generator [5], does not yet solve all of the old problems, but by integrating the generated parser and scanner a significant amount of brittleness associated with developing LALR(1) grammars can be alleviated. This occurs when a context-aware scanner can distinguish between different terminals that have the same or overlapping regular expressions based on the context of the parse. In Copper, this context is the LR parse state and the scanner is designed to only return a token for terminals symbols whose entry in the current state of the parse table are *shift*, *reduce*, or *accepts* — that is, it only will only return a valid token to the parser (or no token in the case of a lexical error in the program).

As an example of the merits of this approach, consider parsing AspectJ, an aspect-oriented extension to Java which introduces new keywords and aspect patterns that pose many challenges to traditional scanning and parsing techniques [1]. The Oxford AspectBench compiler uses a hand-coded moded scanner to deal with these challenges [2]. Bravenboer et al. [1] use scannerless generalized LR parsing to provide the first declarative specification of both the lexical and context-free syntax of the language. By using the context-aware scanner in Copper we also have a fully declarative specification the language but with a deterministic parser and scanner, and we can thus avoid any questions of ambiguity lurking in the grammar [3].

Furthermore, context-aware scanner makes feasible a modular determinism analysis [4] that *guarantees* that the composition of a host language grammar and any set of language extension grammars — which independently pass this modular analysis — will be LALR(1). A goal that is something too often said to be only possible with generalized parsing techniques.

While much of the community is heading in the direction of generalized parsing techniques we feel that there may still be a few worthwhile tricks in the old deterministic approaches.

References

[1] M. Bravenboer, Éric Tanter, and E. Visser. Declarative, formal, and extensible syntax definition for AspectJ. In *Proc. of Conf. on Object-oriented programming systems, languages, and applications (OOPSLA)*, pages 209–228. ACM, 2006.

[2] L. Hendren, O. de Moor, A. S. Christensen, and the abc team. The abc scanner and parser, including an LALR(1) grammar for AspectJ. Available at `http://abc.comlab.ox.ac.uk/documents/scanparse.pdf`, September 2004.

[3] A. Schwerdfeger. *Context-Aware Scanning and Determinism-Preserving Grammar Composition, in Theory and Practice*. PhD thesis, University of Minnesota, Department of Computer Science and Engineering, Minneapolis, Minnesota, USA, 2010.

[4] A. Schwerdfeger and E. Van Wyk. Verifiable composition of deterministic grammars. In *Proc. of Conf. on Programming Language Design and Implementation (PLDI)*, pages 199–210. ACM, June 2009.

[5] E. Van Wyk and A. Schwerdfeger. Context-aware scanning for parsing extensible languages. In *Intl. Conf. on Generative Programming and Component Engineering, (GPCE)*, pages 63–72. ACM, 2007.