Noncanonical Parsing

Sylvain Schmitz [⋆]

Laboratoire I3S, Université de Nice - Sophia Antipolis, France
schmitz@i3s.unice.fr

Abstract. The talk advocates the use of noncanonical parsers wherever deterministic parsers are needed: they preserve the linear time parsing and unambiguity properties, but accept a larger class of grammars.

1 Introduction

Common deterministic parser generators [1] provide a parser developer with two interesting static guarantees: that the input grammar is unambiguous, and that the resulting parser will process its input string in linear time. There is however a major issue with these parser generation algorithms: they cannot provide a deterministic parser for an arbitrary context-free grammar, resulting in the infamous *conflicts* between possible parsing actions. Their inability to deal with parsing decisions that need more than the pre-established k lookahead terminal symbols is to blame for a large part of it.

Two different parsing techniques allow to circumvent this limitation to bounded lookaheads in bottom-up parsers, but to keep the unambiguity guarantee. The first, called regular lookahead parsing, uses a finite state automaton to explore an unbounded right context [2, 3]. The linear time guarantee is however lost. The second, called noncanonical parsing, explores the right context using the parser itself. The latter can thus perform some reductions in this right context, return to the conflict point, and use a bounded number of the newly reduced symbols—representing an entire context-free language—to yield a deterministic decision [4, 5].

We briefly describe a noncanonical LALR(1) parser construction.

2 Noncanonical Parser Construction

Suppose we can generate a canonical LALR(1) parser $\mathcal{C}$ for a given context-free grammar $\mathcal{G}$ [1]. This parser is a pushdown automaton making decisions on configurations of form $[\alpha]\|a$ where $[\alpha]$ is the state reached upon reading the valid prefix α from the initial state $[\varepsilon]$ and a is a single lookahead terminal in the remaining input. Several parsing actions—*shift* or *reduces*—might be possible in a single configuration of $\mathcal{C}$. In such a case a noncanonical parser will shift and explore the right context; it will resume to the conflict point after a reduction and use the reduced symbols to help the decision.

[⋆] Joint work with Jacques Farré and José Fortes Gálvez.

Valid Covers We say that string γ is a *valid cover* in $\mathcal{G}$ for string δ if and only if γ is a valid prefix and $\gamma \Rightarrow^* \delta$. We write $\hat{\delta}$ to denote some cover of δ.

Noncanonical Lookaheads The NLALR(1) parser is able to use nonterminal symbols—resulting from reductions done to the right of the current position—as lookahead symbols. Let us denote by $\mathrm{RLA}([\delta], A \rightarrow \alpha)$ the set of totally reduced lookahead symbols for the reduction $A \rightarrow \alpha$ in canonical state $[\delta]$, and similarily by $\mathrm{DLA}([\delta], A \rightarrow \alpha)$ the set of all lookahead symbols. We define the *conflict* and *noncanonical* lookahead sets for a reduction $A \rightarrow \alpha$ in a set s of canonical states:

$$\mathrm{CLA}(s, A \rightarrow \alpha) = \begin{array}{l}\{X \in \mathrm{DLA}([\delta], A \rightarrow \alpha) \mid [\delta] \in s, X \notin \mathrm{RLA}([\delta], A \rightarrow \alpha), \\ \qquad ([\delta], X) \text{ or } (\exists [\gamma] \in s, X \in \mathrm{DLA}([\gamma], B \rightarrow \beta))\},\end{array}$$

$$\mathrm{NLA}(s, A \rightarrow \alpha) = \Big(\bigcup_{[\delta] \in s} \mathrm{DLA}([\delta], A \rightarrow \alpha) \Big) - \mathrm{CLA}(s, A \rightarrow \alpha).$$

Noncanonical Parser The noncanonical parser is a two-stack pushdown automaton; the second stack acts as an input. We allow a limited amount of backtrack by having reductions push the reduced symbols on top of this second stack instead of the first stack. The states we construct are sets of canonical states $[\alpha]$; let us denote noncanonical states by $[\![\alpha]\!]$:

$$[\![\varepsilon]\!] = \{[\varepsilon]\} \text{ and}$$

$$[\![\delta X]\!] = \{[\widehat{\hat{\gamma} A X}] \mid X \in \mathrm{CLA}([\![\delta]\!], A \rightarrow \alpha), [\hat{\gamma}\alpha] \in [\![\delta]\!]\} \ \cup \ \{[\varphi X] \mid [\varphi] \in [\![\delta]\!]\}.$$

Noncanonical transition from $[\![\delta]\!]$ to $[\![\delta X]\!]$ on symbol X, denoted by $([\![\delta]\!], X)$, exists if and only if $[\![\delta X]\!] \neq \emptyset$. Reduction $([\![\delta]\!], A \rightarrow \alpha)$ exists if and only if there exists a reduction $([\gamma], A \rightarrow \alpha)$ and $[\gamma]$ is in $[\![\delta]\!]$; this reduction uses the noncanonical lookahead set $\mathrm{NLA}([\![\delta]\!], A \rightarrow \alpha)$.

3 Further Developments

The computations for the valid covers and for the noncanonical lookahead sets are expressed as relational equations amenable to efficient processing [5].

More powerful noncanonical parsers with adaptive lookahead lengths are currently considered.

References

1. Donnely, C., Stallman, R.: Bison, The YACC-compatible Parser Generator. The Free Software Foundation (2002)
2. Bermudez, M.E., Schimpf, K.M.: Practical arbitrary lookahead LR parsing. Journal of Computer and System Sciences **41** (1990) 230–250
3. Farré, J., Fortes Gálvez, J.: A bounded-connect construction for LR-regular parsers. In Wilhelm, R., ed.: CC'01. Volume 2027 of LNCS., Springer (2001) 244–258
4. Szymanski, T.G., Williams, J.H.: Noncanonical extensions of bottom-up parsing techniques. SIAM Journal on Computing **5** (1976) 231–250
5. Schmitz, S.: Noncanonical LALR(1) parsing. In Dang, Z., Ibarra, O.H., eds.: DLT'06. Volume 4036 of LNCS., Springer (2006) 95–107