A Predictive Bottom-up Evaluator

Manuel Vilares Ferro* Miguel Angel Alonso Pardo†

Abstract

This paper[1] presents a recursive query processing strategy on Horn Clauses which plays, with respect to classic context-free parsing algorithms, a similar role to the well known LALR compilation schemes. Over a base structure equipped with an efficient context-free parsing algorithm based on dynamic programming techniques, we define a logic push-down automaton which breaks computations into combinable, sharable and storable sub-computations. The latter provides computation sharing and operational completeness and solves most of the problems posed by classic depth-first left-to-right traversals.

1 Introduction

Context-free parsing represents a fine starting point to understand computational aspects of syntactic phenomena in full first order logic based on Horn Clauses. Indeed the Prolog language that popularized them was initially intended as a powerful formalism for describing natural languages based on a context-free backbone [1]. Clear, concise and powerful, they have been largely used for implementing natural languages analyzers [2]. However, there seems to have been few attempts to see whether known theoretical and technical results on context-free parsing (CFP) could be generalized to Horn Clauses, specifically the use of push-down automata (PDA's) and the corresponding form of non-deterministic reasoning [3].

The aim of the work, which is partially reported here, is to devise general purpose algorithmic techniques to deal with formalisms within the continuum that ranges from propositional Horn logic to full first order Horn logic, in practical applications. To attain this goal, we have chosen to work in the context of logic push-down automata (LPDA's) [4], a formal logical engine for the execution of definite clause programs (DCP's) that generalizes the dynamic programming aspects of earlier evaluation strategies. A LPDA is essentially a PDA, in general non-deterministic, that stores logical atoms and substitutions on its stack, and uses unification to apply transitions.

We propose a very simple technique to avoid, in most practical cases, evaluation conflicts by adding lookahead to items in LPDA's following a technique close to that applied by LALR machines in CFP, the lower end of the continuum of Horn-like formalisms. It is possible that not all of the material in this paper is new. We have, however, felt the need for a consistent and coherent exposition.

2 An Informal Overview

The abstract operational model represented by LPDA's allows most of the execution strategies of Definite Clause Programs (DCP's) to unify in a same framework, permitting easy comparison and performance analysis. In relation to this, practical experience has shown that the most efficient methods [4] seem to be those bottom-up approaches including a predictive phase in order to restrict the computation to a useful part of the search space. This relies again on the CFP theory, where the consideration of efficient determinization techniques such as the classic LR ones, differentiate the syntactic contexts in function of the state where the parsing process is, by virtue of the application of a predictive technique in the generation of the parser.

The idea to consider bottom-up evaluation with a top-down predictive control is not new. The Magic Set [5, 6, 7, 8] techniques reduce the number of useless intermediate results produced by using a goal-oriented bottom-up evaluation. The main drawback of this approach is the extra cost these methods require. In effect, the addition of predictive control is made by re-writing logic programs and introducing new predicates. In order to avoid this, we have extended a simple bottom-up evaluation strategy [3] on LPDA's by including a predictive control directly derived from unrestricted CFP [9] in dynamic programming. This control is given, in the simplest version, by a LALR(1) automata, possibly

*M. Vilares is with the Computer Science Department, University of Corunna, Campus de Elviña S/N, 15071 A Coruña, Spain. E. mail: vilares@dc.fi.udc.es.

†M. A. Alonso is currently with the Ramón Piñeiro Linguistic Research Center, Santiago-Noia Road, 3 Km, A Barcia, 15896 Santiago de Compostela, Spain. E. mail: malonso@cirp.es.

[1]partially supported by the **E**ureka **S**oftware **F**actory project, and by the Autonomous Government of Galicia under projects XUGA10501A93 and XUGA20403B95.

non-deterministic, directly computed from the context-free grammar $\mathcal{G}_f$ obtained by considering a parsing rule for each clause in the intensional database. Using the simplest technique this construction is made by keeping only the funtors of the logic terms.

In order to describe the recursive query processing strategy, given a clause $\gamma_k : A_{k,0} : -A_{k,1}, \ldots, A_{k,n_k}$, we introduce:

- The vector $\vec{T_k}$ of the variables occurring in γ_k.
- The predicate symbol $\bigtriangledown_{k,i}$. An instance of $\bigtriangledown_{k,i}(\vec{T_k})$ indicates that all literals from the i^{th} in the body of the clause γ_k, have been proved.

We can now formalize the compilation scheme by the transitions:

$$\forall \gamma_k, \left\{ \begin{array}{llll} 1. & A_{k,n_k} \quad \{E_k\} & \longmapsto & \bigtriangledown_{k,n_k}(\vec{T_k}) \quad A_{k,n_k} \\ 2. & \bigtriangledown_{k,i}(\vec{T_k}) \quad A_{k,i} & \longmapsto & \bigtriangledown_{k,i-1}(\vec{T_k}), \quad i \in [1, n_k] \\ 3. & \bigtriangledown_{k,0}(\vec{T_k}) & \longmapsto & A_{k,0} \end{array} \right.$$

for the reduction mode, and

$$4. \quad \forall \gamma_k,\ A_{k,i}\ \{E_{k,i}\} \longmapsto E_{k,i}\ A_{k,i},\ i \in [0, n_k)$$

for the scanning mode. In each case, $\{E_k\}$ makes reference to the extra control conditions established by the LALR(1) scheme associated to the grammar of functors $\mathcal{G}_f$. So, in the reduction mode, $\{E_k\}$ and $\{E_{k,i}\}$ represent the conditions over the lookahead in the production of $\mathcal{G}_f$ corresponding to γ_k. In the scanning mode, this condition represents the verification of the existence of a term with the same funtor as $A_{k,i+1}$ in order to avoid useless push actions. Briefly, we can interpret these transitions as follows:

1. *Selection of a clause:* When the extra condition $\{E_k\}$ is verified and taking into account the literal A_{k,n_k} on top of the stack, select the clause γ_k whose head is to be proved; then push $\bigtriangledown_{k,n_k}(\vec{T_k})$ on the stack to indicate that none of the body literals has yet been proved.
2. *Reduction of one body literal:* The position literal $\bigtriangledown_{k,i}(\vec{T_k})$ indicates that all body literals of γ_k following the i^{th} have been proved. Now, for all stacks having $A_{k,i}$ just below the top, we can reduce them and in consequence decrement the position literal.
3. *Termination of the proof of the head of clause γ_k:* The position literal $\bigtriangledown_{k,0}(\vec{T_k})$ indicates that all literals in the body of γ_k have been proved. Hence, we can replace it on the stack by the head $A_{k,0}$ of the clause, since it is now proved.
4. *Pushing literals from the database:* The literal $E_{k,i}$ is pushed onto the stack, assuming that literals will be needed in reverse order for the proof.

To shorten the description of transitions we have not detailed the set of possible simplifications, some of them have already been specified in previous works [4] and a lot of them have been derived from the particular strategy considered.

The technique described has also been generalized to a more complex schema by extending the concepts of $first_n$ and $follow_n$ from classic LR parsing to DCP's in a natural manner [9], extending context-free derivation by using unification in a depth-first study of the database. Practical tests have shown that, applying simple prediction schema as LALR(1), the percentage of useless intermediate results not generated is similar to that achieved in corresponding CFP analyzers.

References

[1] A. Colmeraeur, "Metamorphosis grammars", in *Natural Language Communication with Computers. Springer LNCS 63.*, L. Bolc ed., Ed., 1978.

[2] F.C.N. Pereira and D.H.D. Warren, "Parsing as deduction", in *Proc. of the 21st Annual Metting of the Association for Computational Linguistics*, 37-144, Ed., Cambridge, Massachusetts, U.S.A., 1984.

[3] B. Lang, "Towards a uniform formal framework for parsing", in *Current Issues in Parsing Technology*, M. Tomita ed., Ed., pp. 153–171. Kluwer Academic Publishers, 1991.

[4] E. Villemonte de la Clergerie, *Automates à Piles et Programmation Dynamique*, PhD thesis, University of Paris VII, France, 1993.

[5] F. Bancilhon, D. Maier, Y. Sagiv, and J. Ullman, "Magic-set and other strange ways to implement logic programs", in *Proc. of the 5th ACM symp. on Principles of Database Systems*, 1986.

[6] U. Nilsson, "Abstract interpretation: A kind of magic", in *Proc. of PLILP'91*, 1991.

[7] R. Ramakrishnan, "Magic templates: A spellbinding approach to logic programs", in *Proc. of the 5th International Conference on Logic Programming*, 1988.

[8] H. Seki, "On the power of Alexander templates", in *Proc. of the 8th ACM symp. on principle of Database Systems*, 1989.

[9] M. Vilares Ferro, *Efficient Incremental Parsing for Context-Free Languages*, PhD thesis, University of Nice, France, 1992.