# Parsing as Resolution

**Manuel Vilares Ferro**
**Jorge Graña Gil**

Facultad de Informática. Universidad de La Coruña
Castro de Elviña, 76. 15071 La Coruña, Spain

phone: +34-81-100000 Ext. 2068
e-mail: civilare@udc.es

**Abstract**

A general context-free parsing algorithm based on logical dynamic programming techniques is described[1]. The analyzer takes a general class of context-free grammar as drivers, and any finite string as input. In an empirical comparison, the new system appears to be superior to the others context-free analysers[2], and comparable to the standard generators of deterministic parsers[3] when the input string is not ambiguous.

*Key Words and Phrases:* Context-Free Parsing, Dynamic Programming, Horn Clauses, Earley Deduction, Definite Clause Programs, Logical Push-Down Automata.

## 1 Introduction

The problem of context-free parsing is stated in the context of the application of logical compilation schemas to generate very efficient non-deterministic parsers. So, we apply the concept of *logical push-down automaton* (LPDA) introduced by Lang in [3], proposing a new computational paradigm to compile Horn clauses, which is based on a push-down automaton model. By extending to these automata a dynamic programming technique developed for classic push-down automata (PDAs), we obtain a general and simple technique for constructing efficient and complete definite clause programs (DCPs) compilers. In particular, it can be proved that most of earlier proposals as for example Earley deduction [5] or OLDT resolution [6] may be expressed within the general framework of this new formalism.

On the other hand, the use of dynamic programming gives an exponential reduction in complexity forward or backward chaining with most variants of breadth-first or depth-first evaluation, without losing answer completeness.

At this point, context-free parsing is located within the continuum of Horn-like formalisms that ranges from propositional Horn logic to full first order Horn logic. In fact, it can be proved [4] that we can obtain an uniform understanding of the computational aspects of syntactic phenomena within this continuum of Horn-like formalisms, motivated by two facts: Firstly, we can express the computational properties of these formalisms with an unique model, PDAs. Secondly, they use a compatible concept to represent the syntactic structure: the *parse tree* in the context-free case and the *proof tree* in the Horn clause one.

## 2 Context-Free Parsing

The following is an informal overview of parsing by dynamic programming interpretation of LPDAs. Our aim is to parse sentences in the language $\mathcal{L}(\mathcal{G})$ generated by a context-free grammar $\mathcal{G} = (N, \Sigma, P, S)$ according to its syntax, where the notation is $N$ for the set of non-terminals, $\Sigma$ for the set of terminal symbols, $P$ for the rules and $S$ for the start symbol.

### 2.1 The algorithm

We assume that, using a standard technique, we produce from the grammar $\mathcal{G}$ a LALR(1) parser, possibly non-deterministic, for the language $\mathcal{L}(\mathcal{G})$. From here, the technique applied consists in translate reduction schemes in the LALR(1) automaton to clauses in a DCP following a very simple technique described by Vilares in [7]. That allows us to easily represent the dependence of the parsing process on the syntactic context and, from the viewpoint of a future work, the possibility to extent the results to the study of context-sensitive grammars.

On the other hand, being DCPs a generative formalism and not an operational one, the idea is to separate the execution strategy in order to implement several operational approaches to finally choice the definitive kernel of the parser on an experimental basis. In this sense, it can be proved [7, 8] that bottom-up approaches have the best behavior, and that top-down ones are not always correct depending on the representation used for the

[1] this work was totally supported by the **E**ureka **S**oftware **F**actory project.
[2] as for example the SDF system [1].
[3] as for example the standard generator of compilers in UNIX, YACC [2].

stack. To describe this operational formalism, we use the concept of LPDA, in essence a PDA that stores logical atoms and substitutions on its stack, instead of simple symbols, and uses unification to apply transitions.

The algorithm proceeds by building a collection of *items*, essentially compact representations of the stack configurations to guarantee a good sharing quality for the syntactic structures. We associate a set of items $S_i^w$, usually called *itemset*, for each word symbol $w_i$ at the position $i$ in the input string of length $n$, $w_{1..n}$. Items are directly built from literals on a DCP following a given logical compilation scheme that here is a simple variant of the bottom-up one.

New items are produced by applying transitions of the LPDA to existing ones, until no new application is possible. In our case, an item represents the current stack configuration in the form $[p\ A\ q\ S_i^w\ S_j^w]$, where $p$ is the current state in our extended LALR(1) automaton, $A$ is the last recognized symbol, $q$ is the state where we were when the automaton had recognized the symbol $A$, $S_i^w$ is the itemset containing the first token derived from the symbol $A$ and $S_j^w$ is the current itemset.

### 2.2 The shared forest

A major difference with other parsers is that we represent a parse as the string of grammar rules used in a leftmost reduction of the parsed sentence, rather than as a parse tree. However, this difference is only appearance. In effect, context-free grammars can be represented by AND-OR graphs that in our case are precisely the resulting shared-forest directly obtained from the proof shared-forest. In this graph, AND-nodes correspond to the usual parse-tree nodes, while OR-nodes correspond to ambiguities. Sharing of structures is represented by nodes accessed by more than one other node and it may correspond to sharing of a complete subtree, but also sharing of a part of the descendants of a given node. This is a consequence of the binary nature of our transitions, which only depend on the first and possibly second elements stored in the stack. This feature allows us to obtain a cubic space complexity for the shared forest in the worst case.

In this manner, items are used as non-terminals of an output grammar $\mathcal{G}_o = (N_o, \Sigma_o, P_o, S_o)$, where $N_o$ is the set of all items and the rules in $P_o$ are constructed together with their left-hand-side item $I$ by the algorithm. More exactly, we generate an output rule each time a reduce action from the grammar defining the language is applied on the stack. The left-hand-side of this rule is the new item describing the resulting configuration and its right-hand-side is composed by the items popped from the stack in that action, and eventually the number of the reduced rule in the original grammar when we treat the last pop action corresponding to that reduce. So, the start symbol $S_o$ comes from the last item produced by a successful computation.

## 3 Conclusion

The applied logical model emphasizes the role of locality of interpretation with respect to the data structure. In particular, it puts into evidence that the use of states in classic context-free parsing is equivalent to rewriting the original grammar and therefore the sharing quality for the resulting parse structure is affected.

The modularity of the approach emphasizes the possible variations in the operational strategies, showing the importance of the choice of the dynamic framework in relation to correctness and efficiency for the parse process.

Finally, this formalization is quite open to extensions by its conceptual simplicity and it has proved its practical validity in the case of context-free parsing.

# References


[1] **Heering, J. and Klint, P.**
*A Syntax Definition Formalism.*
1987. ESPRIT '86: Results and Achievements, North-Holland Publishing Company, New York, U.S.A (pp. 619-630).

[2] **Johnson, S. C.**
*YACC. Yet Another Compiler Compiler.*
1975. Computer Sciences Technical Report $n^o32$, AT&T Bell Laboratories, Murray Hill, New Jersey, U.S.A.

[3] **Lang, B.**
*Complete Evaluation of Horn Clauses, an Automata Theoretic Approach.*
1988. Research Report $n^o913$, INRIA Rocquencourt, France.

[4] **Lang, B.**
*Towards a Uniform Formal Framework for Parsing.*
1991. Current Issues in Parsing Technology, M. Tomita ed., Kluwer Academic Publishers (pp. 153-171).

[5] **Pereira, F. C. N. and Warren, D. H. D.**
*Parsing as Deduction.*
1983. Proc. of the $21^{st}$ Annual Metting of the Association for Computational Linguistics, Cambridge, Massachusetts, U.S.A (pp. 137-144).

[6] **Tamaki, H. and Sato, T.**
*OLD Resolution with Tabulation.*
1986. Proc. of the $3^{rd}$ International Conference on Logic Programming, London, United Kingdom, Springer LNCS 225 (pp. 84-98).

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

[8] **Villemonte de la Clergerie, E.**
*DyALog. Une Implementation des Clauses de Horn en Programmation Dynamique.*
1990. Actes du $9^{th}$ Séminaire de Programmation en Logique, CNET, Lannion, France (pp. 207-228).