GALENA: Tabular DCG Parsing for Natural Languages*

M. Vilares Ferro† M.A. Alonso Pardo† J. Graña Gil† D. Cabrero Souto‡

Abstract

We present a definite clause based parsing environment for natural languages, whose operational model is the dynamic interpretation of logical push-down automata. We attempt to briefly explain our design decisions in terms of a set of properties that practical natural language processing systems should incorporate. The aim is to show both the advantages and the drawbacks of our approach.

1 Introduction

Logic programming have been extensively used to analyze natural languages [5] due to its declarativeness, the possibility of expressing partial information via variables, the ability of expressing linguistic knowledge as a deductive process and the important role that logic have played in linguistics to represent natural language semantics. Several grammar formalisms have been developed into the realm of logic programming but we restrict this discussions to one of them, definite clause grammars [9] (DCGs), although the results can be applied to other logical formalisms.

DCGs can be considered as a special case of a logic programs and therefore the same techniques applied to the latter could be used to parse definite clause grammars, but performance considerations recommends to follow a specific approach that take into account the intrinsic characteristics of natural languages and the grammars used to describe them. With the development of the GALENA natural language environment we have tried to build a efficient set of tools that could be applied to practical applications and so, in the case of the parser, we have taken the following design decisions:

- Separate lexicon: Natural languages often have a large set of lexical entries with complex inflectional morphology and other formalisms different from DCG, with less formal end experimental complexity, are best adapted to describe them.
- Dynamic programming: In NLP most underlying grammars have undesirable features, such as left recursion or ambiguities, which means that classic top-down parsing with backtracking may not be practical. Dynamic programming guarantees operational completeness and sharing of computations.
- Static analysis: This allows us to reduce the search space in dynamic interpretation. So, both space and time complexity are improved.

*Partially supported by the Government of Spain (project HF96-36) and Xunta de Galicia (projects XUGA10505B96 and XUGA20402B97).

†Departamento de Computación, Facultad de Informática, Universidad de La Coruña, Campus de Elviña s/n, 15071 La Coruña, Spain. E-mail: {vilares,alonso,grana}@dc.fi.udc.es.

‡Centro Ramón Piñeiro para a Investigación en Humanidades, Estrada Santiago-Noia km 3, A Barcia, 15896 Santiago de Compostela, Spain. E-mail: dcabrero@cirp.es.

- Indexation of parse process: Some facilities, e.g. incrementality or error recovery, need some kind of synchronization. Token indexation is a simple technique to obtain synchronization and its application results in a reduction of the search space and make easier the implementation of garbage collection.
- Driving the evaluation: DCGs generalize context-free grammars (CFGs) by extending variables to infinite domains. So, a considerable body of the technology in DCG parsing can be recovered from programming languages.
- Improve Sharing: the level of sharing is a main concern for parsing of natural languages as it has a great impact in the quantity of computations to be performed.

Essentially, GALENA parser can be viewed as an indexed bottom-up adaptation of the *logical push-down automaton* (LPDA) model proposed by Lang in [7], enriched with top-down predictions from a context-free driver in order to filter out undesirable evaluations. In the following sections of this paper, we develop the issues related above.

2 Lexicon

In order to show the complexity of the lexicon of a natural language, we summarize some of the characteristics of a inflectional language like Spanish:

- A highly complex conjugation paradigm, with 3 regular groups and more than 40 groups of irregular verbs. In each group there are 118 inflected forms, without including composed forms. Some verbal forms can have one, two or three clitic pronouns at the end, which often involve changes in the accent position in stems. There are also verb paradigms with gaps and duplicate past participle.
- More than 20 variation groups for gender inflection and more than 10 variation groups for number inflection.

To include a rule in a definite clause grammar for each possible surface form will suppose a great surcharge during the parsing process, degrading the overall performance. Moreover, the developing of lexicon and grammar are often different linguistic activities. To separate them provides independence and allow us to combine different lexicon and grammar sources.

From a performance point of view, instead of using a general mechanism like DCG rules to deal with the specific phenomena of lexicon, it will be convenient to apply specially designed devices like two level rules [10] or finite-state technology [15], which have less computational complexity. We have chosen the latter, compiling a set of morphological rules into a non-deterministic finite-state automaton which allow us to study the segmentation phenomena and to apply verifications strategies based on reductions of the automaton [15]. As an example, an automata that recognize 14,000 Spanish lemmas with 18,000 stems has 230,000 states.

3 Dynamic Programming

Traditional implementations of DCG use the Prolog resolution engine based on backtracking, which presents well known problems with several types of recursive constructions. Instead of backtracking, we apply tabular techniques that store intermediate results in order to achieve

completeness and to avoid redundant computations. In this context, we use LPDA as operational mechanism.

An LPDA is a push-down automaton that stores logical atoms and substitutions on its stack, and uses unification to apply transitions. A parsing strategy when applied to a DCG results in a set of push-down transitions. You does not need to consider the whole stack when a transitions is applied but only a limited context which usually contains the one or two elements on the top of the stack. Bottom-up parsing strategies can fully exploit tabulation as they can take S^1 as dynamic frame [12, 16], that is, they only need to consider the top element of the stack as context in order to apply a transition and therefore only the top elements are tabulated. Each of these top elements forms the core of an *item*.

Generic push-down transitions are of three types. Their application over an item A is given by the following rules, where stacks grow to the left:

- *Swap case:* $(B \longmapsto C)(A) = C\sigma$, where $\sigma = \text{mgu}(A, B)$.

  If the top element A of the stack unifies with B then it is replaced by the element $C\sigma$.

- *Push case:* $(B \longmapsto CB)(A) = C\sigma$, where $\sigma = \text{mgu}(A, B)$.

  If the top element A of the stack unifies with B then a new element $C\sigma$ is pushed in the stack.

- *Pop case:* $(BD \longmapsto C)(A) = \{D\sigma \longmapsto C\sigma\}$, where $\sigma = \text{mgu}(A, B)$.

  If the top element A of the stack unifies with B, then we try to unify the second element with D in order to complete the pop transition, which is equivalent to generate a new swap transition $D\sigma \longmapsto C\sigma$. The composition of these two transitions will result in the replacement of the two elements on the top of the stack by C . Swap transitions generated during the application of pop transitions are called *dynamic transitions* and they are applicable to items resulting from the pop transition, and also to items to be generated.

The LPDA builds items from the initial one applying transitions to existing ones until no new application is possible. An equitable selection order in the search space ensures fairness and completeness, and redundant items are ignored by a subsumption-based relation.

4 Static Analysis

We can consider LPDA-based parsing strategies which are completely guided by the grammar, that is, without considering any information provided by an previous examination of their characteristics. However, experience in context-free parsing shows that information obtained by the static analysis of the grammar allow the parser to gain in efficiency because some useless branches of the parse forest can be avoided and therefore the number of operations to be performed is reduced.

LR parsing techniques are one of the strongest and most efficient class of parsing strategies for context-free grammars and they are based in a finite-state automaton encoded in a set of LR tables containing predictive information which is compiled from the source grammar. These techniques are often applied to deterministic grammars, but considering LR parsing tables in which each entry can contain several actions, we obtain non-deterministic LR parsers [1] which can analyze arbitrary context-free grammars.

As we can think of a DCG as a context-free skeleton with attributes associated to grammatical symbols, we can also apply the same techniques to build a set of LR tables that

encode predictive information about the definite clause grammar under consideration. To do that, we must recover the skeleton, building the set of non terminals from the set of predicates that appear as head of non-lexical rules and the set of terminals from the predicates which are head of lexical rules.

5 Indexation

Synchronization has as advantages to limit the moments in which a transition can be applied and that the input string must be traversed only once [2]. As a consequence, garbage collection techniques can be implemented in order to reclaim items which are not going to be used anymore.

The only disadvantage of synchronization is in the case a infinite number of items must be generated during the analysis of a portion of the input string because the parser will not give any result while a parser without synchronization could obtain the items corresponding to some parse trees but needing infinite time in order to obtain all of them [2]. However, we can put some techniques in practice in order to allow an algorithm with synchronization to detect these kinds of phenomenon [14].

To obtain synchronization, we index the parse by string position. So, we limit the search space at the time of recovery, as parsing progresses, by deleting information relating to earlier string positions. This relies on the concept of *itemset* [6], for which we associate a set of items to each token in the input string, that represents the state of the parsing process at that point of the scan.

We extend itemsets to include dynamic transitions and decrease the number of such structures generated. To do this, it is sufficient to consider itemsets as synchronization points, generating them one by one. So, dynamic transitions in an itemset are only necessary once an empty reduction has been performed and ambiguity arises in the scope of the itemset [12].

6 Driving the evaluation

The result of applying indexation and LR analysis to a LPDA is a set of push-down transitions representing the parsing strategy implemented in GALENA [13].

Items are of the form $[A, i, j, st].\sigma$, where A is a grammar symbol, σ is a substitution, j is the current position in the input string, i is the position in this string at which we began to look for that configuration, and st is a state for an LALR(1) driver controlling the evaluation. Given a DCG with clauses $\gamma_r : A_{r,0} :- A_{r,1}, \ldots, A_{r,n_r}$, we introduce the predicate $\bigtriangledown_{r,l}$ to indicate that all literals from the l^{th} one in the body of γ_r have been proved. We consider the following set of transitions:

$$\text{Reduce}\begin{cases} [A, i, j, st] & \longmapsto & [\nabla_{r,n_r}(\vec{T}_r), j, j, st] \\ & & [A, i, j, st] \\ & & \{\text{action}(st, \text{token}_{j+1}) = \text{reduce}(\gamma_r^f)\} \\ [\nabla_{r,l}(\vec{T}_r), k, j, st_1] & & \\ [A_{r,l}, i, k, st_1] & \longmapsto & [\nabla_{r,l-1}(\vec{T}_r), i, j, st_2] \\ & & \{\text{action}(st_2, \text{token}_{j+1}) = \text{shift}(st_1)\} \\ [\nabla_{r,0}(\vec{T}_r), i, j, st_1] & \longmapsto & [A_{r,0}, i, j, st_2] \\ & & \{\text{goto}(st_1, A_{r,0}^f) = st_2\} \end{cases}$$

$$\text{Shift}\begin{cases} [A_{r,l}, i, j, st_1] \longmapsto [A_{r,l+1}, j, j+1, st_2] \\ \qquad\qquad\qquad\quad [A_{r,l}, i, j, st_1] \\ \qquad\qquad\qquad\quad \{\text{action}(st_1, \text{token}_{j+1}) = \text{shift}(st_2)\},\ \text{token}_{j+1} = A^f_{r,l+1} \\ \\ [A, i, j, st_1] \longmapsto [A_{r,1}, j, j+1, st_2] \\ \qquad\qquad\qquad\quad [A, i, j, st_1] \\ \qquad\qquad\qquad\quad \{\text{action}(st_1, \text{token}_{j+1}) = \text{shift}(st_2)\},\ \text{token}_{j+1} = A^f_{r,1} \end{cases}$$

where A represents any grammar symbol or \$. Expressions between curly braces denote actions, to be tested for the driver before applying transitions, over the context-free skeleton γ_r^f of the clause γ_r. In consequence, our proposal is a bottom-up interpretation scheme profiting from the prediction control due to the LALR(1) driver, the LR technique which is considered best balanced between recognizing power and number of states generated for the finite state automaton.

The first three transitions implement the reduction mode, selecting the clause whose head is to be proved, and applying the corresponding pop transitions to finally push the head onto the stack. The last two transitions correspond to the scanning mode, pushing onto the stack those literals that will be needed for the proof. The parsing process begins with the item $[\$, 0, 0, 0]$ as initial stack element.

Correctness and completeness, in the absence of functional symbols, are obtained from [12, 16], based on these results for LALR(1) parsing and bottom-up evaluation, but it is possible to extend the algorithm to a larger class of grammars. In particular, preliminary work [14] has been developed to deal with cyclic term traversal.

7 Sharing

There exists a close relation between the principle of sharing and the amount of work developed by the parser: sharing of forest reflects the sharing of computations and therefore time and space bounds are affected. However, this relation is not always complete since mechanisms that cause tree duplication are not exclusively grammar dependent.

From the point of view of the dynamic frame the problem consists in determining exactly the amount of information needed by the parser at run-time, which depends on the parsing strategy. So, for example, it can be proved that S^1 is not correct for pure top-down approaches [12, 16].

On the other hand, sharing in the parse forest may correspond to a complete subtree, but also to a tail of a list of sons [3]. Although the use of and-or graphs to represent the shared forest allows this to be dealt with, it does not guarantee an optimal sharing. So, bottom-up parsing may share only the rightmost constituents, while top-down parsing may share only the leftmost ones.

The parsing strategy not only limits the impact of dynamic frames and representation formalisms on sharing, but itself profoundly influences it. Thus, in contrast to grammar oriented approaches, parsers enriched with static predictions show better theoretical performances. In practice, these techniques sometimes lead to distinguishing irrelevant syntactic contexts because a category can be recognized in two different states avoiding sharing. Therefore, in order to achieve the problem to get efficiency a condition has to be found to measure the quality of the parser in relation to syntactic context splitting.

Finally, although the preceding discussion may suggest this, sharing does not exclusively rely on system dependent features, but also on user dependent ones. So, the choice of the parsing

scheme should be in function of the grammar, taking into account corpus to be analyzed. This is, typically, the case of the always difficult adaptation of top-down strategies to left-recursion or cyclic term traversal [11].

8 Conclusion

We have sketched the main theoretical and methodological aspects of a natural language parser based on DCGs, discussing the possibility of efficiently exploiting the formal analysis tools previously developed in context-free theory.

The first goal of GALENA is to optimize performances. In this way, our proposal attempts to be a compromise between sharing of forest and sharing of computations. To obtain this, we have priorized tactical decisions assuring the best sharing, whichever the parsing strategy.

This implies, in the first place, taking a dynamic frame S^1 that provides a maximum sharing quality for stack configurations. Grouping items in *itemsets* referred to the analysis of a same word, and completing sequentially these itemsets, allows for the limiting of the generation of dynamic transitions. So, we can guarantee that a dynamic transition can be used in an itemset if and only if it is not locally deterministic and an empty reduction has been performed on it [12].

The following choice refers to the forest description formalism. The use of and-or graphs allows us to deal with infinite structures and assures the best syntactic sharing for a given parsing scheme and dynamic frame. The problem of tail sharing previously reported is not attributable to the representation chosen, but to the parser. Here, a natural evolution could be the consideration of efficient bidirectional, event-driven parsers [8].

The separation of tagging in a specialized phase reduces, whatever the architecture, parsing costs. To justify this, it is sufficient to take into account the complexity of some inflectional languages shown in section 2. To include lexical information at syntactic level implies an exaggerated increase in the number of predictions, at run-time level for top-down parsers, and statically for those bottom-up ones integrating determinization techniques.

A lot of work has been devoted to debating what is the best parsing strategy in NLP and it seems to be that, for the same deterministic domain, bottom-up architectures are computationally more efficient. As an additional advantage, this allows the problem of infinite prediction loops to be eliminated. So, although the consideration of S^1 limits options to bottom-up parsers, it does not really imply a restriction. Specifically, our main effort is now oriented towards determining the degree of prediction to be integrated in the automaton generation.

At present our bottom-up evaluator is guided by a simple context-free backbone from predicate symbols, which is clearly insufficient, but which permits us to correctly focus the future research. In this sense, preliminary tests have been performed for SLR(k), LR(k) and LALR(k) drivers. Results point to both, LALR as the more adequate approach and the necessity of an extension of prediction to the information contained in arguments. This could be done by extending the well known concepts of `first` and `follow` from CFGs to DCGs, as unification-oriented notions.

In practice, automata generation is totally transparent for the user, at no time cost, and it has been designed as an extension in C++ of BISON [4] to deal with non-determinism and unification, providing efficient array compacting methods to build the tables.

9 Future work

Although it seems our proposal is limited to DCGs with a context-free skeleton close to an LALR(1) grammar, this should be sufficient to deal with formalisms whose ambiguities are well located, as often occurs in practice. Whichever the case, this evaluation guideline can be used for other goals such as efficient cyclic terms traversal.

Several questions are still open. Although we argue that bottom-up parsers improved with the incorporation of predictions are very efficient, it would be desirable to adapt the degree of syntactic context splitting to each grammar, in order to maximize sharing of computations.

This line of research can be further pursued by studying the integration of incremental and error recovery facilities in order to develop highly interactive systems, where the analysis process must be prompted immediately at the onset of new input.

References

[1] M. A. Alonso Pardo, D. Cabrero Souto, and M. Vilares Ferro. A new approach to the construction of Generalized LR parsing algorithms. In Ruslan Mitkov, Nicolas Nicolov, and Nikolai Nikolov, editors, *Proc. of Recent Advances in Natural Language Processing (RANLP'97)*, pages 171–178, Tzigov Chark, Bulgaria, 1997.

[2] François Barthélemy. *Outils pour l'analyse syntaxiqye contextuelle*. PhD thesis, University of Orleans, France, 1993.

[3] S. Billot and B. Lang. The structure of shared forest in ambiguous parsing. In *Proc. 27th Annual Meeting of the ACL*, 1989.

[4] R. Corbett, R.M. Stallman, and W. Hansen. BISON, *Reference Manual*. Free Software Foundation, Inc., 675 Mass Avenue, Cambridge, MA 02139, U.S.A., 1.25 edition, 1996.

[5] V. Dahl. Natural language processing and logic programming. *Journal of Logic Programming*, 19,20:681–714, 1994.

[6] J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[7] B. Lang. Towards a uniform formal framework for parsing. In M. Tomita, editor, *Current Issues in Parsing Technology*, pages 153–171. Kluwer Academic Publishers, 1991.

[8] G. Neumann. A bidirectional model for natural language processing. In *Proceedings of the 5th Conf. of the European Chapter of the ACL*, pages 245–250, Berlin, Germany, 1991.

[9] F. C. N. Pereira and D. H. D. Warren. Definite Clause Grammars for language analysis — a survey of the formalism and a comparison with Augmented Transition Networks. *Artificial Intelligence*, 13:231–278, 1980.

[10] G. Ritchie. On the generative power of two-level morphological rules. In *European Chapter of the ACL*, pages 51–57, Manchester, 1989.

[11] C. Samuelss. Avoiding non-termination in unification grammars. In *Procs. 4th Int. Workshop on Natural Language Understanding and Logic Programming*, pages 4–16, Nara, Japan, 1993.

[12] M. Vilares Ferro. *Efficient Incremental Parsing for Context-Free Languages*. PhD thesis, University of Nice. ISBN 2-7261-0768-0, France, 1992.

[13] M. Vilares Ferro and M. A. Alonso Pardo. An LALR extension for DCGs in dynamic programming. In C. Martín Vide, editor, *Mathematical Linguistics II*. John Benjamins Publishing Company, Amsterdam, the Netherlands, 1997.

[14] M. Vilares Ferro, M. A. Alonso Pardo, and D. Cabrero Souto. An approach to infinite term traversal in DCGs. In *Proc. of APPIA-GULP-PRODE 1997 Joint Conference on Declarative Programming*, pages 307–317, Grado, Italy, 1997.

[15] M. Vilares Ferro, J. Graña Gil, and P. Alvariño Alvariño. Finite-state morphology and formal verification. In András Kornai, editor, *Extended Finite State Models of Language*. Cambridge University Press, 1997.

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