First Results of a French Linguistic Development Environment

L. Bouchard (GIREIL) L. Emirkanian (GIREIL) D. Estival (ISSCO)
C. Fay-Varnier (CRIN) C. Fouqueré (LIPN) G. Prigent (CNET-Lannion)
P. Zweigenbaum (INSERM-U194)

1 Introduction: EGL

The EGL (Environnement de Génie Linguistique) project started in 1989, with the proposal to create a linguistic software development environment containing a computational treatment of French grammar.[1] Its three main objectives were to allow research groups working in NLP:

- to develop and test both general French grammars and specific linguistic analyses for that language,
- to test new parsers and to compare several parsers in a uniform setting, and
- to have at their disposal an analyzer/generator for French, easy to maintain and to port to other domains.

Independently of a particular application, the environment must be usable both as a component in a system making use of an existing syntactic database, and as a development environment for new syntactic treatments of the language. The first phase of the EGL project was partly based on a critical evaluation of existing work (in particular GDE [1]), and defined a general architecture with the following modules:

- parser,
- basic grammar,
- test-suite database,
- lexicon,
- development management tools,
- graphical utilities.

The initial grammatical formalism chosen was that of unification-based grammar and three main linguistic frameworks are taken into account in EGL: GPSG [11], LFG [16] and FUG [17]). The parser is based on the general principle of a chart; different analyzers for the different formalisms can be integrated into the system by making reference to that model and by including specific methods for the types of objects they manipulate. The basic analyzer is a revised version of the GDE parser [8]; two LFG parsers are being integrated, and a FUG parser is planned.

The French test-suite and the grammar are both already fairly well developed. The basic grammar provided with the environment is the keystone of the whole system. It allows using the environment directly and without further work, and also serves as a testbench for the computational solutions to linguistic problems.

The test-suite serves as a guideline for the coverage of (system-provided or user-defined) grammars, to test whether they accept an independently established corpus of written sentences which exemplify the main linguistic problems and phenomena of the language.

[1] The EGL project involves 6 different partners:

- GIREIL: Université du Québec à Montréal, Département Math-Info, Montréal Québec, Case Postale 8888 - Succursale A - H3C3P8, CANADA. <lhb@mips1.info.uqam.ca>, <le@mips1.uqam.ca>
- ISSCO: Université de Genève, 54 rte des Acacias, CH-1227 Genève. <estival@divsun.unige.ch>
- CRIN: Campus Scientifique, BP 239, F-54506 Vandoeuvre-lès-Nancy Cedex. <Christine.Fay@loria.fr>
- LIPN: Université Paris-Nord, F-93430 Villetaneuse. <cf@lipn.univ-paris13.fr>
- CNET-Lannion: Route de Trégastel, BP 40, F-22301 Lannion Cedex. <prigentg@lannion.cnet.fr>
- INSERM-U194: 91 Bd de l'Hôpital, F-75634 Paris Cedex 13. <zweig@frsim51.bitnet>

It was supported by the *Association pour la Coopération Culturelle et Technique* and by the French Programme *PRC Communication Homme-Machine*. Development of the GPSG grammar of French was also supported by grants from the SSRC of Canada (grant #410-89-1469) and the FCAR of Quebec (grants #89-EQ-4213 and #92-ER-1198).

While defining a French lexicon was not one of the main objectives of the project, having a lexicon is an unavoidable requirement for testing grammars and analyzers and the treatment of lexical information became an important component of the work. The need to access a single lexicon required a study of the normalization of lexical information which led to interesting questions about the reusability of syntactic features.

Defining development management tools turned out to pose challenging theoretical problems. The History component keeps track of grammar development and modification, and is complementary to the Coherence component which validates a state of the grammar. The Generation component allows the linguist to test limit cases in the grammar, both from the point of view of analysis complexity and in order to check overgeneration.

We start our description with the module making the system usable as a development tool for linguistic software, i.e. the set of graphical utilities for the visual representation of the grammar, the analysis process and the results.

2 User environment

EGL lets the user parameterize execution and control commands, explore their results, and visualize and edit lexical and syntactic knowledge. In contrast with earlier approaches such as [4], we think that user interface standards are now sufficiently mature to allow reasonably portable software to be developed, and most of these functions are part of a graphical user interface running under X-window Motif. The EGL graphical user interface is best illustrated with the parsing tools, which are directed towards both the grammar developer and the parser developer. The user can select a sentence, control parser execution, and explore the results. During parsing, the user can display the chart and watch it evolve dynamically. The agenda of awaiting chart tasks can also be displayed and manipulated. This allows the parser developer to experiment manually with chart parsing strategies before integrating them into the parser.

After parsing, the grammar developer can display the relevant structures (derivation trees, feature structures, rules used, etc.) and navigate through them. The whole user interface behaves as a structure inspector, or hypertext-style browser, with displays and links tailored to the linguistic needs and habits of the user.

3 Development Management Tools

Besides the test suite elaborated for the project, three validation tools contribute to grammar development: the History, Coherence and Generation components. As the test suite and the History components are described in detail elsewhere [5], we will spend more time on the Coherence and Generation components. They are both based upon a formalism which is common to GPSG, LFG and FUG, and thus able to include all the data and constraints of those three frameworks. In this way, EGL goes beyond previous projects such as [8, 7] and provides a common tool for various frameworks.

A *grammar* consists of four sets (category, (ID-)rule, LP-rule and metarule).[2] Each set includes both data and principles. A principle is a constraint that must apply everywhere and which defines the *admissible* data.

A category (I, F, A) is represented as:[3]

- A categorial identifier I, which is a symbol identifying the category.
- A formula F, which defines constraints applicable to the category. These are deduced from the rule that generated the category, or from principles. The allowed predicates are: standard $\supset$, constrained $\supset_c$, default $\supset_d$ deduction; standard $=$, constrained $=_c$, default $=_d$ unification; negation $\neg$, union $\wedge$ and disjunction $\vee$.
- An attribute-value structure A. A value may be atomic or complex (itself an attribute-value structure). It can be declared explicitly (with constants) or implicitly (referring to another value in the structure, thus allowing data sharing).

Local trees stem from rewrite rules,[4] constrained by LP-rules and principles.[5] The precedence constraints can be mentioned in the right-hand side of a rule *inside* the rule as well as a principle *via* precedence rules. This expressive power (allowing "formalism mixing") facilitates

[2] An (ID-)rule is a regular expression constructed from an Immediate Dominance rule with Linear Precedence constraints.

[3] Each element of a structure or a category can be omitted; in that case, it is considered a variable.

[4] These are themselves defined with metarules.

[5] This is the way to express the Foot Feature Principle, Head Feature Convention, etc. of GPSG.

grammar development. Two examples:

LFG (rewrite rule):

$$(P_{,,}) \rightarrow \{1,(NP, \$0.SUJ = \$1,)\} \wedge \{2,(VP, \$0 = \$2, [TRANS -])\}$$

GPSG (default constraint):

$$(,,[V +, N -]) \supset_d (,,[VFORM V, PASS -])$$

The main problem in the Coherence component is that of *satisfiability*. Is there any valid parse with the user's grammar? Besides satisfiability, some questions are of great interest from a linguistic point of view, e.g. sufficiency and necessity of all the data. A grammar must be *structurally coherent*, and we say that a grammar is coherent iff it satisfies:

- non-cyclicity: there is no cyclic point.
- non-redundancy: A is redundant w.r.t. B in a grammar S iff $S-A$ has the same strong generative capacity as $S-B$.
- non-superfluity: A is superfluous in S iff S and $S-A$ have the same strong generative capacity.
- accessibility-coaccessibility: data is accessible (resp. coaccessible) iff used at least once in generation (resp. a parse).

We have shown [2] that cyclicity, redundancy and superfluity are subproblems of accessibility: an accessibility algorithm can be used as a necessary condition for the three other problems. In a context-free grammar, linguistic coherence can be tested locally. Therefore, a first pass applies to a context-free part of the grammar (without data sharing nor nonmonotonic atomic formulas). A second, global, pass uses label propagation, where labels are defined by constraints. We are also investigating a clique method to treat accessibility in a tractable way [9, 2].

The inputs to the Generation component are the following constraints:

- on the grammar: specification of obligatory, forbidden or cooccurrent rules,
- on terminal nodes: specification of complex structures that determine terminal nodes types,
- on initial structures: specification of incomplete parse trees.

These parameterizations were easily included into the formalism, but problems occur with the algorithm itself, which chart algorithms are insufficient to deal with. Three agendas take care of post-modification of nodes in incomplete trees, thus extending Shieber's algorithm [21, 18].

4 Linguistic Descriptions

4.1 Grammar

The development of the GPSG grammar for French can be traced through three steps.

First, we implemented a demonstration grammar [12], patterned after the English grammar described in the GDE User Manual [8]. In terms of coverage, this French grammar can handle some simple questions, which required the definition of two additional metarules. In terms of grammar-writing style, following a suggestion of [22, pp. 115–119], we define the person feature in terms of two binary features, EGO and PTC (participant). Finally, agreement is a much more pervasive phenomenon in French than in English, and many more cases must be taken into account: adjective/noun, determiner/noun, adjectival predicate, and the past participle.

As a second step, we developed a GPSG-based French grammar along the lines of the English grammar described in [15]. Although the linguistic coverage is similar in both of them, the French grammar is only loosely patterned after the English one. Its development was broken into subtasks according to the types of constituents encountered (AP, NP, VP ...) as well as to the types of specific linguistic problems to be accounted for (e.g. agreement, comparatives and coordination). In general, the rules in our grammar are driven by lexical information: we thus model our computational grammar on the results of current linguistic theory.

Our treatment of agreement is fairly complete [13]. For example, we can handle complex color adjectives (*des robes vert bouteille*, "bottle-green dresses"), predicate APs (*les robes sont vertes*, "the dresses are green"), and past participles (*les étudiantes que les policiers ont matraquées*, "the students that the police beat up").

The treatment of VPs is extensive [14] and includes the positioning of clitics [3] and of negation. Lexical V1 items are used to handle complex tenses and the positioning of negation and certain adverbs. We strived to minimize the number of lexical ID-rules and tackle the problem of "categorial distortion" [20] (in particular, the grammar can account for complement subcategorization alternations in a systematic way).

The treatment of NPs was found to cause

more serious problems. Although we were able to pattern our treatment of modifiers after [15], that of specifiers is more problematic [19]. It has rapidly become clear that semantic information is necessary for a satisfactory solution. Thus, the third step is to enrich our morpho-syntactic grammar with a semantic component [6].

4.2 Lexicon

A lexical database is obviously necessary to perform any test on grammars and parsers. Defining a French lexicon within the GPSG formalism was not one of our goals but, in parallel to the syntactic database, we had to construct a lexicon couched in a formalism compatible with different grammars and with enough coverage to be useful. Like the grammar provided with the environment, this lexicon can be taken as is, or be replaced by the users. We eventually settled on (automatically) transforming the information present in an already existing dictionary (the CNET lexicon) to serve as the lexical database.[6]

4.3 Normalizing Lexical Information

In building a linguistic environment which is both French specific and usable by separate users with independently built systems, we knew that these would require lexical information to be presented in different ways. However, with the assumption that all of the lexical information necessary for the various syntactic analyses is actually present in the lexicon provided with EGL, we make the hypothesis that the content of this information is common to the various systems.

Since an increasing number of grammatical formalisms put a large part of the linguistic description in the lexicon, we are interested in the nature and complexity of lexical entries, in the division of information between grammar and lexicon, in the representation of the syntactic information in the lexicon, as well as in the use of lexical information in the grammar. Normalizing this information thus became an important part of the linguistic aspect of the project: the features in the pre-existing lexicon had to be transformed to serve as the basis for a "neutral" lexicon, which must be usable by grammars not written in the same framework as that of the CNET.

First, a correspondence was established between the syntactic and morpho-syntactic features of the CNET lexicon and the features required in systems created by members of the project: the GIREIL grammar; the LN-2-3 grammar (INSERM); the ELU grammar (ISSCO). From the list of features used by each of them, we extracted those that pertain to the lexicon. We only considered attributes required by the grammars at the lexical level, thus discarding the features which represent information that can only be evaluated during processing, i.e. which cannot be present in a lexical entry (e.g. VEUT-AUX-COMPOSE on a complex verbal form for LN-2-3, or REL on a nominal form in ELU). Since all three systems adopt to some extent a lexicalist approach and include a large amount of syntactic information in the lexicon, this division required a detailed interpretation of their internal workings.

Conversely, although morphological analysis is most often performed in a separate component (i.e. inflected forms do not constitute separate lexical entries), morphological information is included in our normalization, because that information must be present on the lexemes serving as starting points for the syntactic analysis.

We then put in correspondence the lexical features of the various systems; here again, it was necessary to interpret the way they are actually used (e.g. in the representation of reflexive constructions). The normalization of the morpho-syntactic features required in these three grammars can now be extended to other grammatical analyses through the more general list of features established for the mapping which allows each system to recover in the lexicon the information it needs to perform an analysis.

5 Conclusion

While French has been the object of relatively extensive research in computational linguistics, no extensive formal description of that language has been integrated in a linguistically motivated development environment. The EGL project is part of a growing trend towards a wider linguistic coverage coupled with greater flexibility.

Designing a linguistic development environment requires making some fundamental choices about the grammatical formalism, and the evaluation of competing formalisms depends on assumptions imposed by the task at hand (com-

[6] The CNET lexicon has more than 55000 entries defined with 200 keywords. The lexicon is transformed into minimal automata with quasi-linear time complexity for access. The compactness of the automata allows them to be resident in core memory.

plexity, determinism, performance degradation in case of unforeseen input, use and integration of semantic information). The use of NL as a medium for communication between man and machine renders desirable the adaptability of an NLP system to various linguistic formalisms. However, if automatic information processing projects now more often include an NL component, that component is generally "closed" and unmodifiable: few systems are designed to provide the syntactic analysis of natural language texts or to be usable in various contexts.[7] In EGL, several of the modules may be reused outside of the grammatical formalism chosen for our own linguistic description. This basic requirement of system design can have important consequences when we want to tailor the system to applications where the linguistic domain is limited, which is the case in most natural language interface applications. As a design tool, EGL makes it possible to see simultaneously and to manipulate easily each of its components.

References

[1] Baldy, B. and A. de Sousa (1989) *ALVEY : une étude informatique pour la compréhension des mécanismes de l'analyse syntazique azés sur la théorie des Grammaires Syntagmatiques Généralisées.* Rapport de Recherche, LIPN.

[2] Belabbas, A. (1991) *Cohérence des grammaires décrivant le Langage Naturel.* Rapport de DEA, LIPN.

[3] Bès, G. (1988) "Clitiques et constructions topicalisées dans une grammaire GPSG du français". In G. Bès & C. Fuchs eds. *Lexique et paraphrase* pp. 55–81. Lille: Presses universitaires de Lille.

[4] Boguraev, B. J. Carroll, T. Briscoe and C. Grover (1988) "Software Support for Practical Grammar Development." *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, Budapest, pp. 54–57.

[5] Bouchard, L. H., L. Emirkanian, D. Estival, C. Fay-Varnier, C. Fouqueré, G. Prigent and P. Zweigenbaum (1992) "EGL: a French Linguistic Development Environment". Natural Language Processing and its Applications, Avignon 92.

[6] Bouchard, L. H. and L. Emirkanian (1991) *Semantic Interpretation in the Grammar Development Environment.* Working Paper, GIREIL, UQAM.

[7] Carpenter, B. and C. Pollard (1991) "Inclusion, Disjointness and Choice: The Logic of Linguistic Classification." *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, Berkeley.

[8] Carroll, J, B. Boguraev, C. Grover and T. Briscoe (1988) *A Development Environment for Large Natural Language Grammars.* Tech. Report 127, Computer Laboratory, University of Cambridge.

[9] Dechter, R. and J. Pearl (1989) "Tree Clustering for Constraint Networks." *Artificial Intelligence*, 38 (3), pp. 353–366.

[10] Fay-Varnier, C., C. Fouqueré, G. Prigent et P. Zweigenbaum (1991) *Comparaison de systèmes d'analyse syntazique du français : Données et Commentaires.* Journées Nationales du PRC-CHM, Toulouse.

[11] Gazdar, G., E. Klein, G. Pullum and I. Sag (1985) *Generalized Phrase Structure Grammar.* Cambridge: Harvard University Press.

[12] GIREIL (1990a) "Brève description de la grammaire "pochoir" du français". Rapport de recherche. UQAM.

[13] GIREIL (1990b) "Grammaire minimale de l'accord". Rapport de recherche. UQAM.

[14] GIREIL (1991) "La structure du syntagme verbal en français". Rapport de recherche. UQAM.

[15] Grover, C., T. Briscoe, J. Carroll and B. Boguraev (1989) *The Alvey Natural Language Tools Grammar (Second Release).* Tech. Report 162. Computer Laboratory, University of Cambridge.

[16] Kaplan, R. and J. Bresnan (1982) "Lexical-functional grammar: A formal system for grammatical representation". In *The Mental Representation of Grammatical Relations.* J. Bresnan, ed. Cambridge: MIT Press.

[17] Kay, M. (1982). "Parsing in Functional Unification Grammar". In *Natural Language Parsing*, D. Dowty, L. Karttunen and A. Zwicky, eds. Cambridge: Cambridge University Press.

[18] Le Barzic, J.P. (1991) *Génération paramétrée multi-formalisme.* Rapport de DEA, LIPN.

[19] Milner, J.-C. (1978) *De la syntaxe à l'interprétation : Quantités, insultes, exclamations.* Paris: Editions du Seuil.

[20] Milner, J.-C. (1989) *Introduction à une science du langage.* Paris: Editions du Seuil.

[21] Shieber, S., G. van Noord, R.C. Moore, and F.C.N. Pereira (1989). "A Semantic-Head-Driven Generation Algorithm for Unification-Based Formalisms". *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, Vancouver, pp. 7–17.

[22] Tesnière, L. (1988) *Eléments de syntaxe structurale.* (Deuxième édition revue et corrigée. Cinquième impression). Paris: Klincksieck.

[7] The systems developed in France which have been studied in [10] are all concerned with more than the syntactic treatment of the language.