Elimination of lexical ambiguities by grammars

Eric Laporte, Anne Monceaux

ELIMINATION OF LEXICAL AMBIGUITIES BY GRAMMARS. THE *ELAG* SYSTEM

ÉRIC LAPORTE
Université de Marne-la-Vallée - CNRS

ANNE MONCEAUX
Aérospatiale-Matra

Introduction

Due to lexical ambiguity, lexical tagging involves some automatic analysis and recognition of grammatical context, e.g. adjective-noun, subject-verb, etc., in order to check local constraints. Such a short context sensitivity is required, for instance, to recognize technical terms in view of an indexation process, and for context-sensitive spelling checkers. If the lexical tagger is coupled with a syntactic parser, the checking of local syntactic constraints and contextual reduction of lexical ambiguity can considerably speed up the process, by filtering out invalid analyses before parsing (R. Milne 1986). Depending on systems, local constraint checking is viewed either as a part of lexical tagging, since it contributes to selecting tags for words, or as a part of syntactic parsing, since it takes into account contextual constraints, or as an intermediate step. In the case of a lexicon-based tagger like INTEX, a contextual constraint checking module can process the result of the tagger and resolve some lexical ambiguities.

In this article, we present a new, INTEX-compatible formalism of expression of distributional constraints, ELAG (elimination of lexical ambiguities by grammars). We describe the main properties of ELAG and we exemplify them with simple rules formalizing exploitable constraints. We also present an INTEX-compatible evaluation procedure for the lexical disambiguation results.

1. Definition of the problem

For INTEX users, one of the most constant sources of nuisances is the presence of artificial lexical ambiguities in the tags assigned to words during the process of lexical analysis. For example, in the tagging of the following French sentence:

(1) *C'est une véritable marque de fabrique aux yeux des connaisseurs*

the various lexical hypotheses in conflict for the form *marque* include at least a compound noun *marque de fabrique* — the only correct hypothesis in this case —, a form of the feminine noun *marque* and a form of the verb *marquer*. This multiple tagging disturbs the location of linguistic patterns in the text by INTEX: if we try to locate occurrences of constructions like *une marque de parfums* or N_0 *donner une marque d'amitié à* N_1 or N_0 *marquer le livre de la page 30 à la page 36*, sentence (1) might very well show up. This problem comes from the fact that the process of lexical analysis, which is achieved by looking up large coverage dictionaries, retrieves successfully all possible tags of a word, but does not take into account context. We use the term of artificial ambiguities in order to emphasize that these words with several tags are usually not felt as ambiguous in context by the human

reader, as opposed to effective ambiguities, i.e. sentences which have several possible interpretations.

The problem is all the more embarrassing that the tagset is more informative and fine-grained, which is exactly the objective we want to reach by using large-coverage dictionaries and INTEX. Since the average number of tags per form increases with the granularity of the tagset, the quantity of noise in the identification of lexical units in the text increases accordingly.

The general solution to this problem is syntactic parsing, since this operation would, as a side effect, determine the right tag(s) of each word. In some cases, syntactic parsing is even the only way of resolving all lexical ambiguities. In the following sentence, for instance:

(2) *La République ne veut pas que l'école publique, son fleuron, véhicule des signes discriminatoires,*

the part of speech of *véhicule* cannot be determined without a thorough recognition of the syntactic structure of the sentence, nor without syntactic information about the verbs *vouloir* and *véhiculer*, namely the fact that they are transitive verbs with a direct complement.

However, in many cases, a simple constraint involving basic linguistic information about words in the immediate context suffices to resolve much of the artificial lexical ambiguity in a text. For example, if we modify sentence (2) by inserting *ne* to the left of *véhicule*:

(3) *La République ne veut pas que l'école publique, son fleuron, ne véhicule des signes discriminatoires,*

the ambiguity of *véhicule* is resolved in favour of the verb, since in French, the form *ne* can be followed by various parts of speech, but never by a noun.

This type of effect can be identified as a realistic objective of a system of resolution of lexical ambiguity. We can express it as follows:

- increasing the precision in the tagging of the words, i.e. removing as many as possible of incorrect analyses, which does not imply that we hope to eliminate all of them,
- without reducing the recall, i.e. with the strict constraint of never discarding a correct analysis.

The principle of a system of lexical ambiguity resolution is to obtain this effect by exploring only a local context of the words marked as ambiguous after dictionary lookup.

A trade-off between recall and precision may be suited for some applications, but not for the most fundamental of them, which is syntactic parsing. In this article, the objective of never discarding a correct analysis is considered as a strict constraint. In order to increase the precision as much as possible, we can focus in priority on the lexical ambiguities that contribute the most massively to lexical ambiguity in sentences.

Given these objectives, the best approach should be to handcraft and maintain directly the distributional data required for the reduction of lexical ambiguity. These data are elaborated by taking into account underlying linguistic structures, by abstracting general rules from observable facts, and by expressing them in a readable form. During the past ten years, several systems following this scheme have been presented (M. Silberztein 1991; E. Roche 1992; T. Vosse 1992; H. Paulussen and W. Martin 1992; A. Voutilainen 1994; Ph. Laval 1995). This scheme is opposed to another, in which the distributional data is automatically acquired by frequency-based corpus training (I. Marshall 1983; F. Jelinek 1985; R. Garside 1987; J. Benello et al. 1989; D. Hindle 1989; D. Cutting et al. 1992; E. Brill 1992; H. Schmid 1994; E. Roche and Y. Schabès 1995).

Building a grammar of resolution of lexical ambiguity is a challenge: correct analyses

should not be removed; the results of syntactic parsing cannot be explicitly used, since it is not available at the time when ambiguity resolution is applied to the text; the linguistic analysis that we wish to assign to a sentence must be taken into account, which implies that the writer of a grammar of resolution of lexical ambiguity has particular views about the desired results of syntactic analysis; and finally, a formalism is indispensable: it should be as simple as possible, but rules cannot be expressed in the form of statements in natural language. In the following, we expose how ELAG helps facing these issues.

2. Tagset

ELAG is compatible with INTEX, and the set of lexical tags that can be attached to words is the INTEX tagset. These tags appear in various forms in INTEX files and windows. In ELAG grammars, they appear as e.g. <*faire*.V:Kms>, which comprises:

- the canonical form or lemma of the lexical unit. In <*faire*.V:Kms>, this form is *faire*, whereas the inflected form described by the tag is *fait*. The inflected form occurs in the text and is not explicitly represented in the tag. The canonical form may be a compound;
- the part of speech, here *V* for verb;
- if the part of speech allows for it, a series of inflectional feature values, here *Kms* for past participle, masculine, singular. When an inflected form is ambiguous between several series of inflectional values for the same canonical form, for example <*faire*.V:Kms> and <*faire*.V:P3s>, each of these separate tags is a complete tag; the format <*faire*.V:Kms:P3s> is an abbreviation;
- immediately to the right of the part of speech, the lexical tag can include a subcategory, like in <*émission de télévision*.N+NDN:fp> where NDN indicates the internal structure of the compound.

The lexical information conveyed by the tag depends on the dictionary. Since such tags comprise all the information stored in the dictionary, we call them complete tags. Complete tags are used in the representation of tagged texts. In ELAG, they can also appear in rules formalizing distributional or grammatical constraints which are specific for a particular word. For example, let us state formally that *aussi* cannot be tagged as a coordinating conjunction (CONJC) when it is followed by a sentence boundary. We will assume that INTEX recognizes sentence boundaries and that the dictionary describes *aussi* as (at least) an adverb:

Luc est aussi arrivé à l'heure

and as a CONJC:

Luc s'est dépêché, aussi est-il arrivé à l'heure

and we will write our first rule. An ELAG rule always comprises an "if" part and one or several "then" parts. We can formulate our example as:

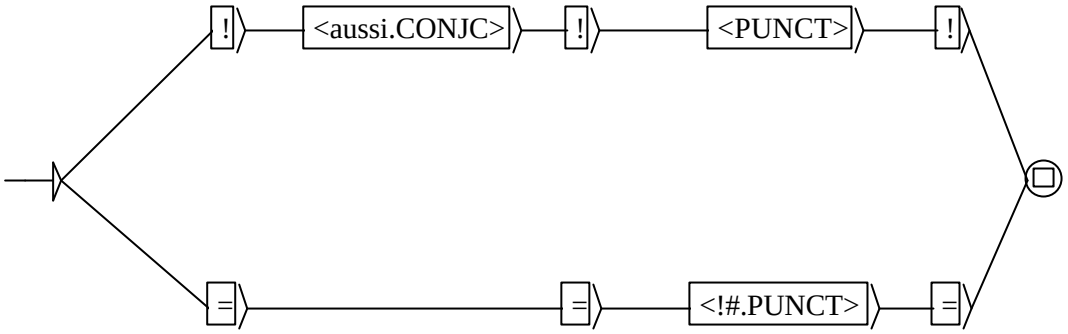
If <*aussi*.CONJC> is followed by a punctuation,
then the punctuation cannot be a sentence boundary.

In order to express this constraint formally, we will use the symbol # which represents sentence boundaries. Other non-verbal symbols are represented by their actual form: , / ' and - all are considered as tags; <PUNCT> is a variable that stands for any of these symbols, and <!#.PUNCT> stands for any <PUNCT> except #. With this notation, our constraint becomes:

If <*aussi*.CONJC> is followed by <PUNCT>,

then this <PUNCT> is <!#.PUNCT>.

Thus, the "if" part describes the pattern <*aussi*.CONJC> <PUNCT>, and the "then" part describes a single <!#.PUNCT>. Practically, the "if" part is always signalled and delimited by three boxes with "!", and each "then" part by three boxes with "=", and the rule looks like Graph 1.



Graph 1.

The boxes with "!" and "=" are delimitors which are used to recognize the structure of the rules. All other boxes contain linguistic elements that are searched in input sentences when rules are applied to text. The left and right "!" and "=" are present to make the rule more readable. The central "!" and "=" are the only element of synchronization between the "if" part and the "then" part. As a matter of fact, when the rule of Fig. 1 is applied to a sentence, ELAG checks that whenever <*aussi*.CONJC> is immediately followed by <PUNCT>, the juncture between them is immediately followed by <!#.PUNCT> in every analysis, and it removes the analyses that do not obey this constraint. When the form *aussi* ends a sentence, the CONJC tag will be correctly removed because it violates the constraint, but the ADV tag will correctly remain unchanged because it is not concerned by the rule.

The "if-then" structure of ELAG disambiguation rules is borrowed from M. Silberztein (1993), because it is a very natural way of formulating readable disambiguation rules.

The recognition of the context usually involves the use of tags for categories of words, e.g. parts of speech, like <N>, which stands for any noun. We call them variable tags because they represent a set of usual lexical tags. The conventions of formalization of variable tags are an important element of the formalism of description of grammatical constraints. In variable tags, the part of speech can be combined with:

- one or several inflectional values: <N:s> for any noun in the singular, <V:1s> for any verb in the first person singular,
- or with a canonical form: <*fait*.A>, <*fait*.A:f>, etc.,
- or with one or several canonical forms and exclamation marks, making up a negative variable tag, e.g. <!*être*!*avoir*.V> which represents any verb except *être* and *avoir*.

When an inflected form is ambiguous between several series of inflectional values for the same canonical form, for example <*faire*.V:Kms> and <*faire*.V:P3s>, the format <*faire*.V:Kms:P3s> can be viewed as a variable tag or as an abbreviation for a list of complete tags.

Graph 1 makes use of variables standing for categories of symbols: <PUNCT> and <!#.PUNCT>. They are also variable tags, since non-verbal symbols are considered as tags.

These conventions are slightly different from current INTEX conventions for formalizing

grammatical patterns to be located in texts[1]. In particular, the Locate menu does accept a type of variable tags entirely natural, but not accepted by ELAG: tags reduced to a single word, like *fait*, which represent any complete lexical tag that the dictionary may associate to the form. In the case of *fait*, there are at least four of them: <*fait*.N:ms>, <*faire*.V:Kms>, <*faire*.V:P3s>, <*fait*.A:ms>.

In fact, our conventions about variable tags impose that any tag should include at least a part of speech, like <*fait*.A:fs>, <*fait*.A>, <A:fs>, except the joker or universal tag <> that represents any tag[2]. With this convention, if we formalize a constraint to resolve a lexical ambiguity with respect to the form *est*, we are not allowed to refer to it through the variable tag *est*, so we will necessarily resort to lexical tags with a part of speech, like <*être*.V:P3s> or <*est*.A:ms>.

Generally, writers of descriptions consider any restriction to the expressive power of the formalism as an obstacle to their work of elaboration and of generalization from their intuition, because it restricts the technical means provided by the formalism to deal with some category of grammatical constraints. However, paradoxically, a restriction of the expressive power of the formalism may make the activity of formalization easier. This is the case of this convention:

- In case of a doubt about the tags associated to a form in the linguistic analysis underlying the system, the convention incites the rule writer to refer to the electronic dictionary and to take into account its contents.

- There is no reason why contexts relevant to the distinct tags — the verb and adjective, for example — should have anything in common. When the rule writer describes contexts relatives to one of them, he needs not take into account the other one, not even the ambiguity of the form. Describing grammatical constraints is usually more comfortable when one can ignore completely the possible ambiguity of the elements being described. The paradox is only apparent, since the existence of ambiguity is the reason why undertake the construction of grammars, but what is relevant to the description is not the ambiguity but the distributional and syntactic contexts of the specific linguistic elements themselves. Moreover, during filtering, analyses are processed as if they were entirely distinct.

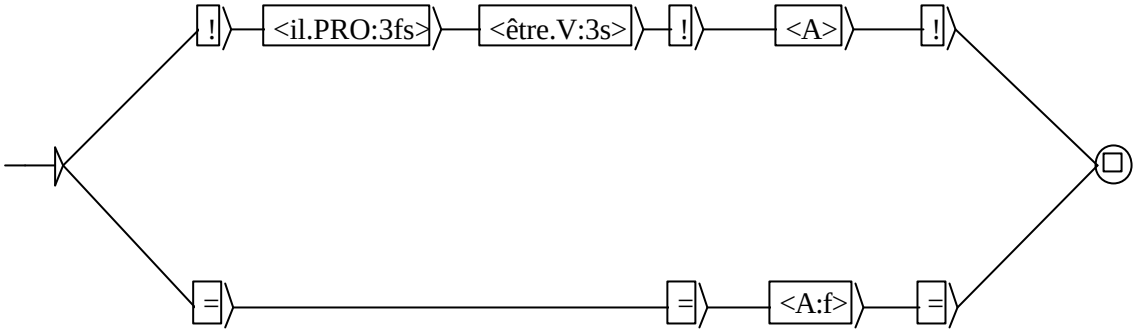
- Given that grammatical constraints are expressed with tags which comprise at least a part of speech, they refer to specific grammatical constructions, and sometimes to definite word senses. This is why the correctness of the rules remains unchanged in case one introduces into the dictionary new lexical tags describing new acceptions of the same words. Consider, for instance, the following sentence:

Elle est fidèle à ses idées

and a grammatical constraint designed to resolve the gender ambiguity of the adjective:

[1] For example, at the time of our experimentations, the Locate menu of INTEX did not accept variable tags with both a canonical form and a grammatical category.

[2] PUNCT is considered as a part of speech. The set of unknown words, i.e. the forms not described in the dictionary, is also considered as an additional part of speech: <?>.



Graph 2.

Now, imagine that we introduce into the dictionary of proper nouns a tag <*Elle*.N+pr:ms> for a magazine titled *Elle*. The rule of graph 2, when applied to:

(4) *Elle était plein de reportages cette semaine,*

does not apply to the correct analysis with <*Elle*.N+pr:ms>, and therefore does not reject it. If we write another version of graph 2 by substituting a variable tag *elle* for <*il*.PRO :3fs>, i.e. without complying to the convention being discussed, the new version becomes erroneous with the introduction of <*Elle*.N+pr:ms> into the dictionary: when it is applied to (4), it eliminates all analyses with <*plein*.A:ms> and retains only the analyses in which *plein* is tagged differently. Therefore, the correct analysis is discarded.

We can conclude that our convention reduces the degree of interdependency between the contents of the dictionary and of grammars. More precisely, the use of graph 2 does not dispense with introducing the new tag into the dictionary, since otherwise graph 2 rejects the correct analysis of *plein* in (4); but at least, graph 2 does not need any modification when the new tag is introduced, because it is specific enough.

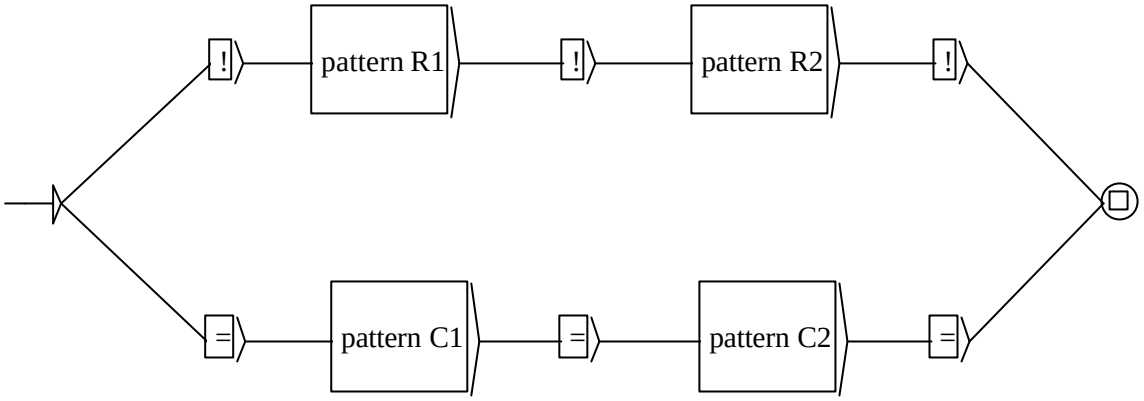
This paradoxical situation, in which a limitation of the expressive power of the formalism makes easier the use of the formalism itself, is well known in the field of software engineering. Object-oriented programming is a particularly well suited practice for creating and maintaining complex software systems. One of the means successfully used in this framework to facilitate the implementation of programs is to prevent the implementer from using certain variables in certain conditions.

3. Specification of the effects of applying ELAG rules

Graphs 1 and 2 give a rough idea of the effect produced by the application of a rule to a sentence. In this section, we will specify this effect in a more detailed and general way.

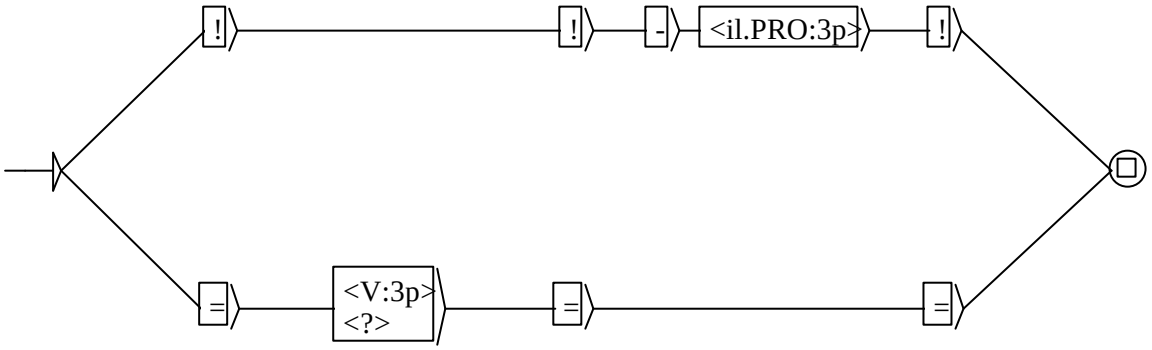
Any ELAG rule consists of an "if" part delimited by boxes with "!" and one or more "then" part delimited by boxes with "=". The boxes with "!" and "=" are delimitors whose only function is to mark the structure of the rules. All other boxes contain linguistic elements that are searched in input sentences when rules are applied to text.

The simplest form of an ELAG rule comprises only one "then" part. The boxes with "!" and "=" delimit four patterns R_1, R_2, C_1, C_2, as in graph 3. Any of these four patterns can be the empty word.



Graph 3. General form of an ELAG rule with one "then" part.

The constraint expressed by such a rule is that whenever an occurrence of R_1 is immediately followed by an occurrence of R_2 in an analysis of a sentence, then the juncture between them must be immediately preceded by an occurrence of C_1 and followed by an occurrence of C_2 in the same analysis. The effect of such a rule is to remove all analyses that do not obey this constraint. For example, graph 4 states that when a pronoun *ils* or *elles* follows a dash, the word before is always a verb in the third person plural.



Graph 4.

This graph was written assuming that the pronouns *ils* and *elles* are described in the dictionary by the tags <*il*.PRO:3mp> and <*il*.PRO:3fp>. The two lines in pattern C_1 mean that the word before the pronoun must be either a <V:3p> or an unknown word <?>. An unknown word is a form which is not found in the dictionary. The tag for unknown words was included here in order to avoid the situation where the verb before the pronoun would not be in the dictionary. Should that happen, graph 4 would correctly not eliminate the analysis <?>-<*il*.PRO:3p>. In this graph, patterns R_1 and C_2 consist of the empty word, which means that the corresponding context is not taken into account. This rule resolves partially or totally the lexical ambiguity of the verb: if it is ambiguous with an adjective, like *convergent*, or with a <V:3s>, like *pressent*, these tags are correctly discarded.

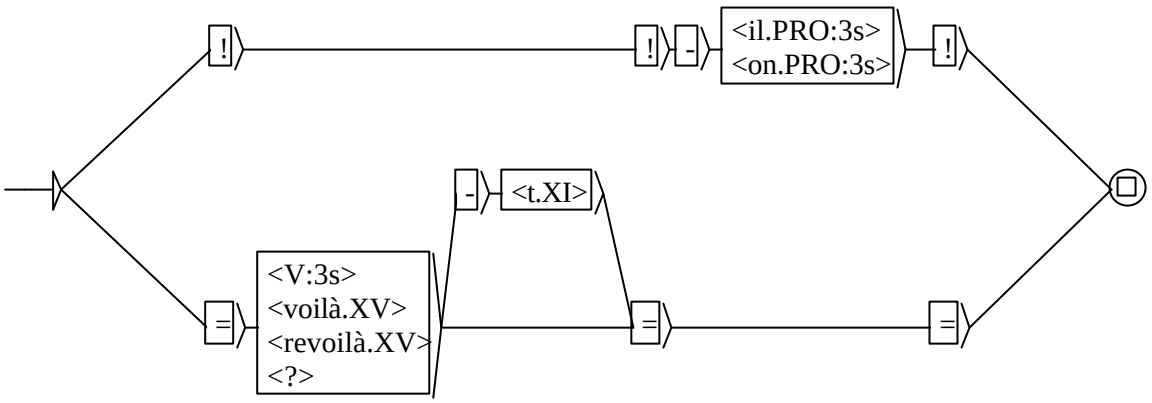
There is no restriction on the lengths of the sequences that belong to patterns R_1, R_2, C_1, C_2. In fact, any of these patterns represents a set of tagged sequences that may not have all the same length. For example, let us write an equivalent of graph 4 for the singular. In addition to the <V:3s>, we will have to take into account the form *-t-* which is inserted between the verb and the pronoun in the case of some verbs — but not all:

Avait-il l'intention de se manifester ? Le fera-t-il maintenant ?

If we wish to tag *-t-* as <*t*.XI>, where XI is a dummy part of speech for this type of isolated phenomena, we can write the rule as in graph 5. In this graph, pattern C_1 contains various sequences of tags; some of them comprise three tags, others one. The graph states that if the pronouns in the "if" part are preceded by a dash, then the dash is preceded by one of the patterns in the "then" part. Graph 5 contains other additional elements as compared to graph 4:

- <*on*.PRO:3s>. If we omitted <*on*.PRO:3s> from the "if" part, the rule would become less general, but would still be correct.

- <*voilà*.XV> and <*revoilà*.XV>. If we omit these complete tags from the "then" part, and if *voilà* and *revoilà* are not tagged as verbs in the dictionary, the rule will discard the correct analysis of *voilà-t-il.*



Graph 5.

Graphs 1, 2, 4 and 5 exemplify a practical feature of ELAG which is important for writers of disambiguation grammars: what is described in rules are sequences that can occur in correct analyses of correct texts. This feature distinguishes ELAG from the system of E. Roche (1992), which asks the user to describe sequences of tags that can never occur in a correct analysis of a correct text. One of the reasons for us to design ELAG was the opinion of several potential grammar writers, which found that describing correct sequences would be more comfortable and easier than describing incorrect sequences.

The constraints expressed in graphs 1, 2, 4 and 5 are all very simple, but it is usually necessary to include more complex patterns in order to formalize more elaborate constraints. The context explored during the application of a rule is usually very local, but a given rule may describe rather various contexts, and the formalism does not set any bounds to the length (in number of tags) of the sequences described in contexts. The context explored by ELAG is thus local, but not bounded. This feature distinguishes ELAG from frequency-based taggers, which usually explore a context of a fixed length, often one or two words.

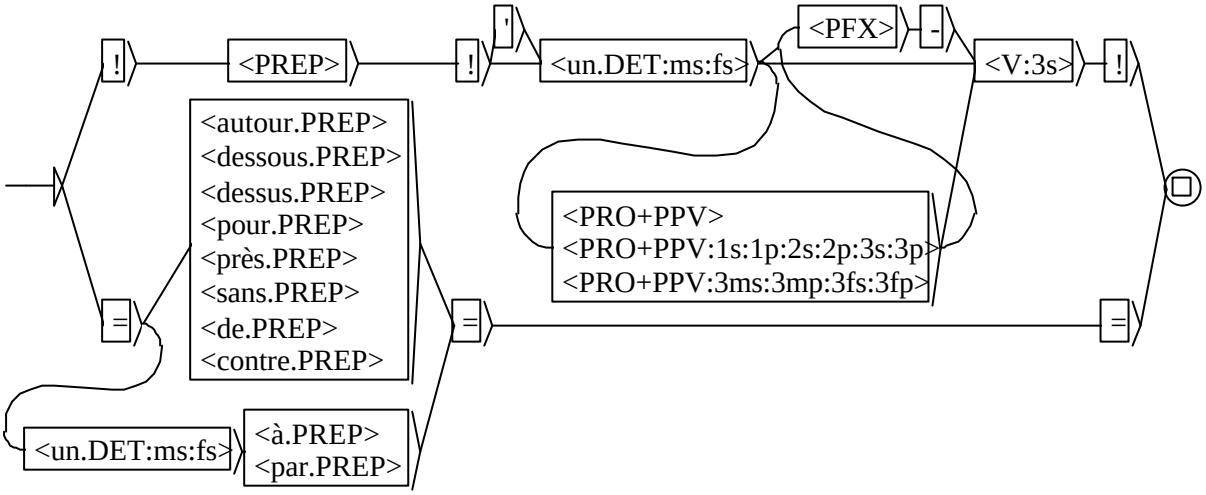
In addition, patterns R_1 and C_1, or patterns R_2 and C_2, may describe portions of text which overlap partially or totally. In graph 6, patterns R_1 and C_1 overlap one another. This graph assumes that in the syntactic analysis that underlies the whole system (dictionary, ambiguity resolution and syntactic parsing), the determiner *un* with a deleted noun should still be tagged as determiner:

Plus d'un réussira

The graph states in which conditions this determiner can occur between a preposition and a <V:3s>. The rule applies even if a preverbal pronoun (PPV), or a prefix (PFX) with a dash, or both, occurs between *un* and the verb. This graph resolves lexical ambiguities of the verb: any analysis in conformity with the "if" part but not with the "then" part is rejected. This rule discards correctly, for example, the tag <*préfacer*.V:3s> in:

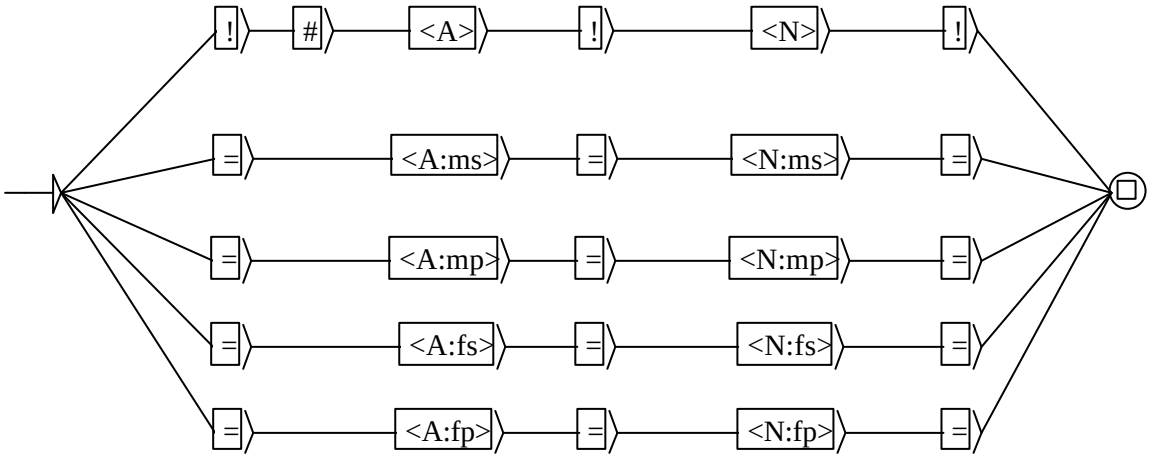
Le livre commence par une préface

In this graph, patterns R_1 and C_1 overlap one another. In the case of prepositions *à* and *par*, C_1 explores a little more than R_1 to the left. In general, there is no reason why the "if" part and the "then" part of a rule should cover the same portion of text. Graphs 4 and 5 exemplify this.



Graph 6.

The most general form of an ELAG rule includes several "then" parts, meaning that whenever the "if" part is recognized, at least one of the "then" parts should be present. For example, graph 7 states that a sequence <A> <N> after a sentence boundary agrees in gender and number.



Graph 7.

The "then" parts of this graph do not check for the presence of the sentence boundary,

because the "if" part does, and a second check would not change the constraint expressed by the rule. This graph is simple, but imprudent. In fact, in the beginning of a sentence, an adjective in the plural can be followed by a coordination of several nouns in the singular:

Chers Marie et Alexandre, acceptez ce clin d'œil

If we apply graph 7 to this sentence, it will discard the correct analysis. The graph should be modified in order to avoid this effect.

Examples are useful for understanding a formalism intuitively, but they are not equivalent to a rigorous specification. Let us now formulate in mathematical notation the constraint expressed by a rule.

The simplest case is when the rule comprises only one "then" part, and therefore 4 patterns R_1, R_2, C_1, C_2. Each of these patterns is a finite automaton representing a set of sequences of tags. The tags in the patterns may be complete tags or variable tags, but a variable tag represent a set of complete tags, so in any case each pattern represents a set of sequences of complete tags (including punctuation tags and unknown-word tags). Note that each sequence is of finite length, but the set of sequences may be infinite, since there is no bound to the lengths of the sequences. Technically, each pattern represents a rational set on the alphabet A of complete tags. Each pattern can be reduced to the empty word, but not to the empty set.

Let P be the set of taggings of a sentence. Each tagging of the sentence, i.e. each element of P, is a sequence of complete tags. Note that all taggings in P do not necessarily have the same length, because of compound words. We will specify the constraint, and therefore the effects of applying the rule to the sentence, by defining the set of taggings in P which satisfy the constraint. To do this, let us examine first which sequences do not satisfy the constraint: in such sequences, either C_1 is absent:

$$(A^*R_1 \setminus A^*C_1)R_2A^*$$

(where "*" means iteration and "\" means set subtraction), or C_2 is absent:

$$A^*R_1(R_2A^* \setminus C_2A^*)$$

or both. Consequently, the set of taggings in P which satisfy the constraint is:

$$(5) \quad P \setminus (A^*R_1(R_2A^* \setminus C_2A^*) \mid (A^*R_1 \setminus A^*C_1)R_2A^*)$$

where "|" means union.

If there are n "then" parts, we need a more general formula to specify the constraint. Let us name C_1^1, C_2^1, C_1^2, C_2^2..., C_1^n, C_2^n, the $2n$ patterns in the n "then" parts. If a sequence does not satisfy the constraint, there is at least a subset I of [1, n] such that all C_1^i are absent from the sequence for each i in I, and such that all C_2^i are absent for all i in [1, n] \ I. Hence, the generalization of (5) is:

$$(6) \quad P \setminus \bigcup_{I \subseteq [1,\, n]} (A^*R_1 \setminus \bigcup_{i \in I} A^*\, C_1^i)\, (R_2A^* \setminus \bigcup_{i \in [1,\, n] \setminus I} C_2^i\, A^*)$$

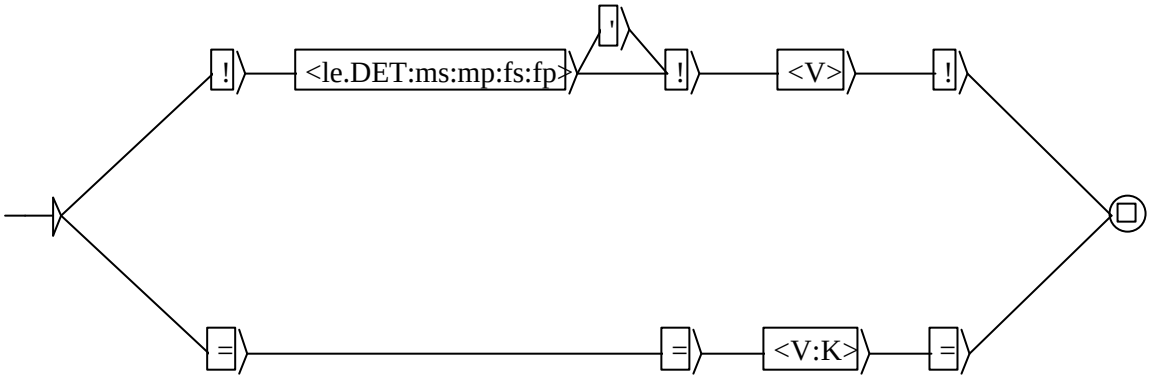
Expressions (5) and (6) show an important feature of the formalism: P appears only once in these expressions. In other words, the set of analyses that do not satisfy the constraint (and, consequently, the set of analyses that satisfy the constraint) is independent of the set of analyses to which the rule is applied. This feature may look a bit theoretical, but in fact it has

important consequences on the practical use of the formalism.

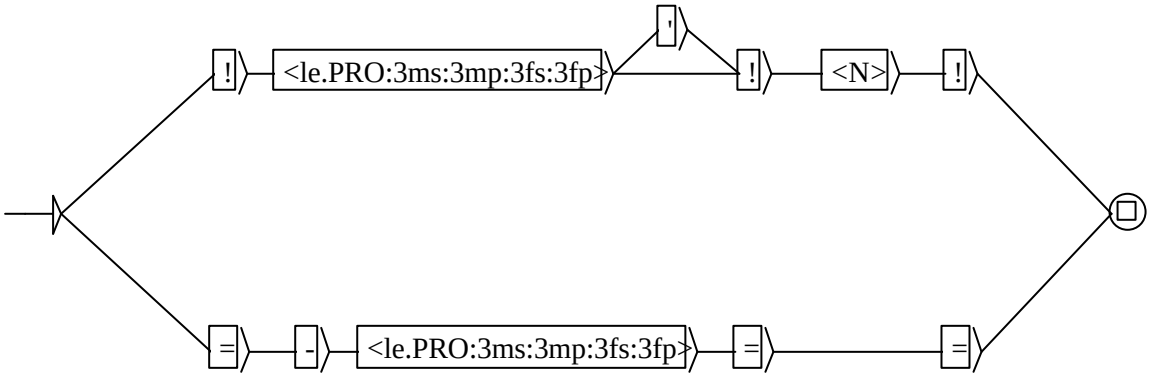
4. Independence of ELAG constraints

The various graphs examined above are very simple examples, and more complex graphs are needed for more complex constraints. However, there is a limit to the complexity of ELAG rules: they must be simple and compact enough to be readable by writers of disambiguation grammars. Practically, the size of an ELAG rule is limited to a page. Given that dozens of constraints can be exploited in order to resolve lexical ambiguity, the only possible strategy for building a disambiguation grammar is to accumulate separate rules and to use them together, in order to benefit from the addition of their effects.

In the preceding section, we specified the effect of applying an ELAG rule. What is the effect of applying a disambiguation grammar made of several ELAG rules? The answer is very simple: the constraint formalized by a set of rules is a logical *"and"* of the individual constraints formalized by the rules. An analysis of a sentence is accepted by the set of rules if and only if it would be accepted by each rule used in isolation. For example, consider graphs 8 and 9, which are adapted from E. Roche (1992).



Graph 8.



Graph 9.

Graph 8 states that the determiner le is not followed by a verb, except in the past participle. Graph 9 states that the pronoun le is not followed by a noun, except if it is preceded by a dash. If we apply both rules to the following sentence:

(7) *Les banques l'expédient dans un délai de quatre à six semaines,*

graph 8 correctly eliminates the analyses with

<*le*.DET:ms:fs> ' <*expédier*.V:P3p:S3p>

and graph 9 those with

<*le*.PRO:3ms:3fs> ' <*expédient*.N:ms>,

but the analyses with

<*le*.DET:ms:fs> ' <*expédient*.N:ms>

and

<*le*.PRO:3ms:3fs> ' <*expédier*.V:P3p:S3p>

are accepted, because they satisfy the constraints of both graphs. Graph 8 applies only to the analyses with the verb, and graph 9 only to those with the noun, as if each analysis were processed in an entirely distinct way.

In addition to ELAG, several other systems of resolution of lexical ambiguity use logical "*and*" to combine rules (E. Roche 1992; A. Voutilainen 1994). In Ph. Laval (1995), one of the two formalisms proposed (p. 101) is of the same type.

Such systems, including ELAG, have a remarkable property: grammatical constraints expressed as separate rules are independent of each other and do not show any interactions. A given rule has a specific effect on texts, and this effect will be the same, no matter whether this rule is used with other rules or not. When you add a new rule to an existing grammar, the formalism itself ensures that the effects of the existing grammar are not modified. The new rule can only reject further analyses. In particular, this implies that when you introduce new rules to a grammar, the rate of reduction of lexical ambiguity can increase but never decrease.

In addition, the order of application of rules is indifferent: a set of rules can be used either simultaneously or sequentially, and in any order, without any modification of the final result. If the same set of rules is applied again to the result, or even applied iteratively, the result remains unchanged after the first application. This comes from the fact that constraints are combined with logical "*and*", a commutative operation.

The independence of constraints and the independence of analyses are two fundamental properties of such systems. They make them easy to build and to maintain. The effects of disambiguation rules being cumulative, rules can be written separately and used together, even without any coordination between rule writers[3]. The only risk with this procedure is to duplicate work: there cannot be compatibility problems. However, the price to be paid for this flexibility is a limitation of the expressive power of these systems. For example, the following rule can be stated in English but cannot be directly translated in ELAG:

(8) In any sequence of the form <*avoir*.V> <V:K>, all the lexical ambiguity of the first word is resolved in favour of the verb *avoir*, and all the lexical ambiguity of the second is resolved in favour of the past participle.

This is a priority rule. Let us apply it to the following sentence:

(9) *Nous avions relevé l'incident le jour même*

[3] Provided they are based on the same linguistic analyses.

The tags <*avion*.N:mp> and <*relevé*.N:ms> are correctly rejected, but this correct application of the rule depends on the presence of the tags <*avoir*.V:I1p> and <*relever*.V:Kms> in other analyses for the same words. Now let us examine how rule (8) behaves in the presence of other rules. If another rule had previously (incorrectly) eliminated <*avoir*.V:I1p> or <*relever*.V:Kms>, the application of (8) would leave the sentence unchanged, since its conditions of application would not be satisfied. Therefore, the result of the application of rule (8) depends on whether other rules have been applied to the same sentence before. In addition, the final result of the application of a set of rules depends on the order of application of the different rules. Rule (8) cannot be expressed in ELAG, since ELAG does not provide any means of describing a constraint about an analysis through the description of another analysis. Of course, the lexical ambiguity of compound tenses in French can be partially resolved with the aid of other grammatical constraints.

In other examples of grammatical constraints that cannot be formalized in ELAG, the description of the context takes into account lexical ambiguity, as in rule (10):

(10) When a form is ambiguous between <V:P3s> and <N:fs>, like *aide*, and occurs before an <A:fs>, the ambiguity is resolved in favour of <N:fs>.

For instance, when this constraint is applied to:

Nous avons besoin d'une aide urgente

it correctly rejects the tags <*aider*.V:P1s:P3s:S1s:S3s:Y2s>, but this correct behaviour depends on the presence of <*aide*.N:fs>, which is another tag for the same word, and therefore belongs to other analyses. The result of the application of (10) depends on other rules, since they may have eliminated <*aide*.N:fs> before (10) applies.

Let us now exemplify a type of rule in which a form is recognized only if all its lexical ambiguity has been resolved beforehand. Rule (11) is a more prudent variant of (8):

(11) In any sequence of the form <*avoir*.V> <V:K>, if all the lexical ambiguity of the first word has already been resolved in favour of the verb *avoir*, then all the lexical ambiguity of the second is resolved in favour of the past participle.

This version may correctly accept the tag <*relever*.V:Kms> in (9), but only if another rule has already rejected <*avion*.N:mp>. Therefore, there may be implicit dependencies between rules in the grammar. Rule (11) cannot be expressed in ELAG, since there is no way of describing how a constraint on a given analysis depends on the absence of other tags for one of the words.

The independence of rules, in ELAG and in related formalisms, excludes the existence of a network of rule/exception relations between the rules. For example, in graph 5, the two forms *voilà* and *revoilà* are the only exceptions to rule (12):

(12) Any word occurring to the immediate left of *-t-il*, *-t-elle* and *-t-on* is a verb.

If we decide to adopt (12) as a rule, and to state separately that rule (13):

(13) *voilà* and *revoilà* may occur to the immediate left of *-t-il*, *-t-elle* and *-t-on*.

is an exception to (12), we introduce explicit dependencies between rules. The effects of (12) and (13) cannot be determined independently of each other.

The independence of rules have several other important practical consequences. If a given analysis is eliminated by a set of rules, this implies that at least one of the rules eliminates the analysis, no matter in which order they are applied. Practically, if the application of a disambiguation grammar rules out a correct analysis, a situation which we want to avoid, we can apply each of the rules to the sentence separately, and we are sure to discover which of the grammars reject the correct analysis. Then, if we modify these rules in order to correct them, these modifications will not change the effects of other rules. This feature makes it possible to detect precisely the origin of errors, and to stick to the objective of never discarding a correct analysis during the evolution of the disambiguation grammar.

It is possible to define what a correct disambiguation rule is, by checking that all the analyses ruled out are incorrect, with reference to the underlying linguistic analyses on which one wishes to build the system. The way of combination of grammatical restrictions ensures that a set of correct rules is correct.

The description of grammatical constraints with this type of systems is also comfortable for a practical reason: rule writers can completely ignore the possible lexical ambiguities of the elements they are describing, since the distributional and syntactic contexts of linguistic elements are recognized independently of other constructions that could happen to be homographic with them.

5. Overlap between application zones

In ELAG, the context relevant to the processing of a case of lexical ambiguity is explicitly defined by the rule writer, and the formalism ensures that a larger context does not become implicitly relevant.

When rules are applied to a text, the recognition of sequences selects portions of text that we will call application zones. For example, the application zone of graph 8 is always two words, with possibly an apostrophe between them. Given the variety of situations of lexical ambiguity, a disambiguation grammar generally contains numerous rules; due to the density of lexical ambiguity in texts, the application zones of rules in a text frequently overlap each other, i.e. contexts relative to adjacent elements may partly coincide. For example, in the following sentence:

(14) *Sans doute plus d'un y a-t-il déjà pensé*

the application zones of graphs 5 (*a-t-il*) and 6 (*d'un y a*) overlap each other. Two distinct application zones of the same rule can even present such an overlap, if the beginning and end of the contexts described in the rule contain some identical part. An application zone can even be completely embedded in another.

In such cases, the ELAG formalism ensures that the effect of applying each rule to the relevant application zone is independent of the rest of the text, to the left as well as to the right. In (14), both graphs 5 and 6 apply in the same way as they would if each of them were used alone. This feature facilitates the construction of a disambiguation grammar by accumulating rules. The effects of each rule is independent of all others: they are not altered with the introduction of new rules, they are justed added to the effects of the new ones. In addition, the extension of the context relevant to each rule remains strictly local, and does not change from what was decided by the rule writer.

6. Evaluation

The evaluation of the results of lexical disambiguation is indispensable in order to be able to

compare versions, for monitoring the evolution of a disambiguation grammar, and to compare different systems, for determining the relative position of systems in the competitive domain of lexical disambiguation. Unfortunately, the performances of systems with different tagsets cannot be compared without a specific work on the two tagsets.

Once a given tagset has been chosen, a comparison between grammars or between versions becomes theoretically possible.

In addition to ELAG, we implemented a separate, INTEX-compatible procedure of evaluation of the results of lexical disambiguation. This procedure can be used with another procedure of disambiguation than ELAG, provided that the input and output of the system are available in the form of finite automata, and more specifically in .fst format, which is the main format in which INTEX generates tagged texts. The evaluation involves two independent parametres: recall, i.e. the proportion of analyses accepted by the system among correct analyses, and precision, i.e. the proportion of correct analyses among accepted analyses. In current literature, these parametres are usually not evaluated separately. However, in our view, the status of recall and precision is very different, since we consider as a priority to ensure that a correct analysis is never rejected. Therefore, we will consider separately recall and precision.

6.1. *Evaluation of recall*

The problem is to detect cases of correct analyses being eliminated during the resolution of lexical ambiguity. This work cannot be automated, otherwise we would dispose of an automatic procedure able to determine the correct analyses of sentences in unrestricted text, which is not the case yet. On the contrary, the procedure is nearly entirely manual. In practice, we use two types of computer aids.

a) A readable listing of the text after processing allows us to manually examine unrestricted text, and to check whether the correct analyses are present. We implemented a listing format in which the tagged text is displayed in lines. A file in this format is generated from the .fst output of the system of lexical disambiguation. The automaton editor of INTEX, FSGraph, could also be used, provided that a file in .grf format can be generated from the .fst output of the system.

b) A complementary method consists in automatically detecting sentences in which the system of lexical disambiguation rejects all analyses. This detection is easy, because the output for these sentences is empty. These sentences are usually not very numerous, but it is worth examining their a priori analyses. These sentences fall into three cases:

- either the text itself is incorrect, like

Les atteintes au paysages (sic),

a title with an agreement mistake;

- or the correct analyses do not belong to the automaton of the sentence after dictionary lookup, e.g. because a lexical item is absent from the dictionary, like in

Je me suis bien amusé, au revoir, et merci !

where *au revoir* belongs to a class of compounds which cannot be reliably recognized by INTEX yet and has not been integrated into the dictionaries;

- or the correct analyses are present in input and therefore are rejected by the grammar.

6.2. *Evaluation of precision*

A rigorous evaluation of precision involves determining the correct analyses of sentences, since precision can be defined as the proportion of correct analyses among accepted analyses. This work cannot be automated on unrestricted text for the same reason as before, but another parametre, the rate of reduction of lexical ambiguity, is computable and gives a good evaluation of precision provided that recall is high enough. The rate of reduction of lexical ambiguity can be defined as the proportion of lexical ambiguity removed during the application of the grammar. The quantity of lexical ambiguity is roughly defined as the number of tags per word. If this quantity is r_1 in input and r_2 in output, the reduction rate is simply $(r_1 - r_2)/ r_1$.

In current literature, the quantity of lexical ambiguity is usually defined as the average number of tags per simple word. This definition is simplistic because it makes this quantity insensitive to a certain type of resolution of lexical ambiguity. For example, graphs 8 and 9, when applied to sentence (7), reject approximately one half of the analyses of *l'expédient*, but do not reject any of the tags of any of these two words. They only reject combinations of these tags. Doing this, they remove a part of the lexical ambiguity of the sentence, but the usual definition of the quantity of lexical ambiguity does not measure this difference. This imperfection comes from the fact that this type of resolution of lexical ambiguity can only be implemented if output is represented in the form of acyclic finite automata, whereas standard disambiguators use less powerful means of representation of their output.

We propose a definition of the quantity of lexical ambiguity that takes into account the removal of paths in an automaton even if the number of tags per simple word remains unchanged. We cannot define the quantity of lexical ambiguity as, for instance, the number of states or transitions of the automaton divided by the number of simple words. Indeed, the number of states or transitions of an automaton can either increase or decrease when one removes paths, even if states or transitions are counted in the minimal deterministic automaton. Instead of this, we substitute the geometric mean for the arithmetic mean when we compute the average number of tags per simple word. If the i-th simple word has a_i tags, the geometric mean r is defined by:

$$\log r = (\log a_1 + \log a_2 + ... + \log a_n) / n \qquad (15)$$

The numerical result is not very different from the arithmetic mean $(a_1 + a_2 + ... + a_n) / n$, but (15) can be generalized to the case of an acyclic automaton. If any tag of a word can be combined with any tag of the next word, the number of paths of the automaton is $a_1a_2...a_n$. We can therefore generalize (15) to the case of any acyclic automaton, by computing the number p of paths of the automaton:

$$\log r = (\log p)/n$$

With this definition, the quantity of lexical ambiguity is close to the value given by the standard definition, but more sensitive to partial resolution of lexical ambiguity.

This definition is implemented in our procedure of evaluation of redution rate. The user of this procedure obtains for each sentence and for the whole text a rate that represents the proportion of ambiguity removed during the processing.

7. Implementation

The implementation of ELAG follows formulae (5) and (6). For simplicity, we will comment only about (5) which is a particular case of (6). In (5), P depends only on the text and the rest of the expression depends only on the rule. Since the same rule is used on many texts, as

much as possible of the computation is executed independently of *P*, the result being an automaton *R* which is the compiled form of the rule:

(16) $R = A^* \setminus (A^*R_1(R_2A^* \setminus C_2A^*) \mid (A^*R_1 \setminus A^*C_1)R_2A^*)$

The automaton of *P* is generated by INTEX. If only one rule is to be applied to *P*, the result is obtained by an automaton intersection between *P* and *R*:

(17) $P \cap R$

If several rules are to be applied together, the compiled forms of these rules are submitted to automaton intersection and the result is the compiled form of the grammar. It is applied to a text *P* by (17).

Conclusions and perspectives

Approximately 70 ELAG rules for French have been written and tested on texts. The examination of residual lexical ambiguity in the output of these tests shows that existing rules can be extended and numerous new rules could be written and tested. In fact, the possibilities of construction of lexical disambiguation grammars are still largely unexplored.

An extension of the set of variable tags is now indispensable for writers to design more elaborate ELAG rules. In their present state, the conventions for variable tags do not take into account subcategories, e.g. +pr in <N+pr> for proper nouns, and this restriction is felt as an obstacle to the design of efficient rules.

Finally, ELAG is not quick enough to process big texts with big grammars. A time optimization of the program of application of rules to texts is under study.

Authors' addresses

Éric Laporte
IGM, Université de Marne-la-Vallée - CNRS
5, bd Descartes
F-77454 Marne-la-Vallée CEDEX 2
France
eric.laporte@univ-mlv.fr

Anne Monceaux
Aérospatiale
12, rue Pasteur
B.P. 76
F-92152 Suresnes CEDEX
France
anne.monceaux@aeromatra.com

ACKNOWLEDGMENTS

The implementation of ELAG came after a few years of reflection and work on resolution of lexical ambiguity. Éric Laporte began to get interested by this question in 1992. He designed a first version of the formalism in 1996 (with one "then" part). Anne Monceaux and he wrote the first rules in 1997, some of them in imitation of grammars by Max Silberztein (1993) and Maurice Gross (1997). The software was implemented in 1997 by Éric Laporte and his

undergraduate class of computer science at the University of Rheims, in compatibility with INTEX for NextStep. The second version of the formalism (with *n* "then" parts) was designed in 1998 by Éric Laporte and Ali Issa. They adapted the software to INTEX for Windows and to the second version of the formalism in 1999. We thank all those that showed their enthusiasm for this project and those that contributed to it, even without any direct participation. This work was partially financed by the European Union project Copernicus 621 Gramlex.

REFERENCES

Benello, J.; A.W. Mackie; J.A. Anderson. 1989. Syntactic category disambiguation with neural networks. *Computer Speech and Language* 3:203-217.

Brill, Eric. 1992. A simple rule-based part-of-speech tagger. In *Proceedings of the 3d Conference on Applied Natural Language Processing*, Trento, ACL, pp.152-155.

Cutting, D.; J. Kupiec; J. Pedersen; P. Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the 3d Conference on Applied Natural Language Processing*, Trento, ACL, pp.133-140.

Garside, R. 1987. The CLAWS word-tagging system. In *The Computational Analysis of English*, Garside and Sampson (eds.), London/New York: Longman.

Gross, Maurice. 1997. The construction of local grammars. In *Finite-State Language Processing*, E. Roche and Y. Schabès (eds.), Cambridge, Mass./London, England: MIT Press, pp. 329-354.

Hindle, D. 1989. Acquiring disambiguation rules from text. In *Proceedings of the 27th Annual Meeting of the ACL*, pp. 118-125.

Jelinek, F. 1985. Self-organized language modelling for speech recognition. In *Impact of Processing Techniques on Communication*, J. Skwirzinski (ed.).

Laval, Philippe. 1995. Un système simple de levée des homographies. *Lingvisticae Investigationes* XIX(1):97-105.

Marshall, I. 1983. Choice of grammatical word-class without global syntactic analysis: Tagging words in the LOB Corpus. *Computers and the Humanities* 17:139-150.

Milne, R. 1986. Resolving lexical ambiguity in a deterministic parser. *Computational Linguistics* 12(1):1-12.

Paulussen, H.; W. Martin.1992. Dilemma-2: a lemmatizer-tagger for medical abstracts. In *Proceedings of the 3d Conference on Applied Natural Language Processing*, Trento, ACL, pp. 141-146.

Roche, Emmanuel. 1992. Text disambiguation by finite-state automata, an algorithm and experiments on corpora. In *Proceedings of COLING-92*, Nantes.

Roche, Emmanuel; Yves Schabès. 1995. Deterministic part-of-speech tagging with finite-state transducers. *Computational Linguistics* 21(2):227-253.

Schmid, F. 1994. Part-of-speech tagging with neural networks. In *Proceedings of COLING-94*, Kyoto.

Silberztein, Max. 1991. A new approach to tagging: the use of a large-coverage electronic dictionary. *Applied Computer Translation* 1(4):23-39.

Silberztein, Max. 1993. *Dictionnaires électroniques et analyse automatique de textes. Le système INTEX*, Paris: Masson, 233 p.

Vosse, T. 1992. Detecting and correcting morpho-syntactic errors in real texts. In *Proceedings of the 3d Conference on Applied Natural Language Processing*, Trento, ACL, pp. 111-118.

Voutilainen, A. 1994. Morphological disambiguation. In *Constraint Grammar: a Language-Independent System for Parsing Unrestricted Text*, F. Karlsson, A. Voutilainen, J. Heikkilä, A. Anttila (eds.), Berlin/New York: Mouton-de Gruyter.