A non-recursive sentence segmentation, applied to parsing of linear complexity in time

Jacques VERGNE

Jacques.Vergne@univ-caen.fr

LAIAC Université de Caen F-14032 Caen cedex France

Abstract

First, we intend to show that parsing is often seen as a combinatorial problem, for some reasons which are not due to properties of natural languages.

Secondly, we describe a method for building a syntax, that we apply to build the foundations for a syntax of natural languages.

Thirdly, we apply these syntactic foundations to build a non-combinatorial parser, therefore one of linear complexity in time.

Keywords: syntax, parsing, linear complexity

Introduction: research domain

This work takes place within a fundamental research on natural language syntax.

Syntax is seen as a science of a real object: texts written in a natural language, and also seen as a study of forms, irrespective of meanings.

Our method is scientific and experimental: studied objects are corpora[1], with manual observations, computer aided observations (with statistics) and experiments, to find syntactic regularities and build a theory which models them.

Computers and parsing make up an experimental device, an observation tool of the object, and a modelling tool of the theory, which enables us to make linguistic hypotheses and evaluate their validity.

Parsing methods and strategies are by-products of linguistic concepts: we make the hypothesis that the better linguistic concepts are modelled, the better the parser works.

This work is mainly about French[2] on both aspects: linguistic study and parsing. A precise linguistic study has also been done about Spanish[3] and English[4]. Studies about the transposability of concepts have been done on German, Polish, Basque, Japanese (with speakers of each natural language) and Latin. Otherwise proven, these concepts seem to be general properties of natural languages.

I. Is parsing really a combinatorial problem?

Parsing is usually seen as a combinatorial problem; it implies that the best algorithms are of quadratic complexity. We want to show here that it is not due to properties of the processed material (a natural language), but to non-appropriate linguistic concepts and parsing strategies.

I.1 An usual definition of parsing

Parsing a sentence usually involves: tagging and linking words, delimiting and identifying phrases, assigning syntactic functions to phrases, and outputting all possible analyses (zero or more) according to a formal grammar.

I.2 A definition of a combinatorial problem

A combinatorial problem is a problem where we have to choose attributes of objects (categories of words), but we do not have any criterion to directly make an *individual choice* (this word has this category), we only have criteria to make a diagnostic on the whole solution: this chain of words is or is not a sentence according to a set of word tags and to a formal grammar. *Individual choices* are then made at random, and if a choice is bad, another is tried (that

[1] Every example in this paper comes from a corpus.

[2] The French corpus is made up of two informative (scientific) texts: the preface of a book about pattern recognition, and a paper in a review about marine biology (7000 words, 250 sentences).

[3] In collaboration with Eduardo López Gonzalo and Luis A. Hernández Gómez of the E.T.S.I. Telecomunicación, Universidad Politécnica Madrid, on a corpus (1700 words, 80 sentences) from an economics review (see [López 93], chapter 4, pages 115 to 169).

[4] The English corpus is made of a paper in the same review about marine biology (2200 words, 100 sentences).

is usually called backtracking); this implies a tree process where none or many solutions are found.

I.3 Why are parsers usually combinatorial?

Three factors make parsers combinatorial (only one of them would be enough). They mainly lie in non-appropriate linguistic concepts leading to non-appropriate parsing strategies.

I.3.a Word polycategory

A given written form may have different categories in different contexts. The most frequent polycategory is between noun and verb. Polycategory occurs in context but is usually not solved in context, with context-free grammars: all possible categories are put in the dictionary (is it possible to do?) and will be tried while parsing.

I.3.b The linguistic concept of phrase

Chomsky defined the concept of phrase in a theory of competence, without having concern for parsing, which has to deal with real texts, therefore performance. He defined the concept of phrase as comprising its complements, which themselves are phrases. This recursive definition amounts to define at the same time a segment (phrase) and a dependency (a complement-phrase depends on a phrase), and to say that this dependency is marked by contiguity. For instance, the following segment is a single phrase (which comprises a NP which comprises a NP):

(the Council (for the Exploration (of the Sea)))
• <--------- • <------- •

A recursive segment makes the segmenting problem combinatorial, because it mixes segmenting and linking from both linguistic and algorithmic points of view.

I.3.c The use of formal grammars

In parsing, a formal grammar has two functions at the same time: it guides the parsing process which chooses a rule to apply, and it also validates phrase patterns; these two functions may be separated in two more specialised tools.

The use of formal grammars makes parsing combinatorial because there are more than one applicable rule at a given time, at different places in the sentence, without any criterion to choose which rule at which place. In formal grammars, phrases are modelled in recursive rules, which have to be applied an unknown number of times, without any criterion to choose among them.

I.4 How to build a non-combinatorial parser?

Let us make the hypothesis that it is possible to build a non-combinatorial parser because non-semantic ambiguity is an artefact.

A non-combinatorial parser should output the only right analysis, because it should have all the criteria to make all choices.

I.4.a Word polycategory

Word polycategory emerges from a lexical view of parsing: parsing usually begins with consulting dictionaries, in which all words with all their possible categories are supposed to be collected. However, we can see parsing as a contextual resolution, where contexts and positions of words in a sentence are more important than consulting dictionaries.

We use the solution either to tag words by a contextual deduction from some written forms (preposition, determiner, ...), or to give a single default category coming from a morphological study.

I.4.b Non-recursive segments

Let us conceptually separate segmentation and relation: let us define segments without using relations (this is a fundamental principle).

By defining non-recursive segments (and validating them by corpora studies), we remove this cause of "combinatorialness". It implies a real segment hierarchy between word and sentence, and allows a global parsing strategy by climbing up in the segment hierarchy; this has the advantage of simplifying and specialising every level: segments of a level are defined and delimited in terms of segments of the level below, and relations between segments of a level are within a segment of the upper level. Once a level is parsed, larger segments are delimited and will be the base for parsing the upper level.

Non-recursive segments also allow to segment first, and to link after, thus algorithmically separating these two problems; segmenting alone is non-combinatorial, and linking segments within an upper level segment is positional, then non-combinatorial; only some "free" segments as "prepositional phrases" cannot (now) be linked by positional clues.

I.4.c A propagation of local deductions

It is possible to tag nearly every word by propagation of local deductions: from the categories of some contiguous words, it is possible to make totally sure deductions on the categories of some next words in a sentence.

These deductions rest on linguistic properties (therefore non-probabilistic), and are independent of these next words; it means that a dictionary is not

necessary to tag many words, mainly adjectives and nouns, the category of which can be deduced by pure positional means.

II. Syntactic foundations

We will first describe an approach for building a syntax of natural languages, and then present the syntactic foundations in the same three stages:

II.0 How to build a syntax of natural languages ?

Building a syntax of natural languages involves three successive stages, mainly for conceptually separating segmentation from linking:

1) Defining a hierarchy of non-recursive segments

In such a hierarchy, every segment is non-recursive, therefore made up of segments of **another** type than itself: a segment of the level below.

While climbing up in the segment hierarchy, *sequences* are made up of words, *blocks* are made up of sequences, *sentences* are made up of blocks, and so on with paragraphs, sections, chapters and books.

This hierarchy of segments is to be considered in total generality from word to text, through the sentence and the paragraph (see [Lucas 92] on paragraph structure, and [Lucas 93] on book structure). The present work concerns this hierarchy between word and sentence, and is to be placed in a wider research between word and text.

Segment structures are described in terms of segments of the level below: sequence structures in terms of words, block structures in terms of sequences, sentence structures in terms of blocks.

Segment positions and order are described within a segment of the level above: words positions within a sequence, sequences positions within a block, blocks positions within a sentence.

Particular attention is given to the topology of the linear chain, seen as a one-dimensional space: segments hierarchy, topological relations of contiguity or "insertion" between segments of the same hierarchic level. At this stage, these segments are defined according to their place in the hierarchy, their relative positions at a given level, their structures in terms of the segment below, but no linguistic relation (complementation, subordination, or dependency) is used to define them (it would imply a recursive definition): segmentation and relation are conceptually separated. Of course, segments are defined according to multilingual corpora studies.

2) Defining a linguistic relation between these segments

Once segments are defined, it is possible to define a linguistic relation between them at every level: dependency between two segments:

- dependencies between two words within a sequence are given by words positions in the sequence
- dependencies between two sequences within a block are given by sequences positions in the block
- dependencies between two sequences of different blocks within a sentence are not directly given by sequences positions in the sentence, but are constrained by topological properties of the dependency tree (see II.2.c below).

3) Defining a segment made up of contiguous linked segments

Contiguous dependent sequences make up a new segment from sequences (1) and dependencies (2): the chain of contiguous dependent sequences.

II.1 Defining a hierarchy of non-recursive segments

Between words and sentences, we observe 2 levels of non-recursive segments:

- nominal sequences and verbal sequences are made up of contiguous words; sequence structures are defined in terms of words; relations inside sequences are relations between words; sequences are in a topological relation either of *contiguity* or *discontiguity* with other sequences;
- blocks are made up of 1 to 3 contiguous sequences (in French); block structures are defined in terms of sequences; relations inside blocks are relations between sequences; blocks are in a topological relation either of *contiguity* or *discontiguity* or *surrounding-insertion*. with other blocks.

Thus, a sentence can be seen at these two levels:
- as a chain of contiguous sequences
- as a chain of contiguous or inserted blocks

II.1.a Nominal sequences, verbal sequences and clips

The term "sequence" is chosen to emphasise contiguities around the noun or verb, to stress the nominal-verbal symmetry (on both linguistic and algorithmic aspects) and also to distinguish it from a (recursive) phrase:

a nominal sequence is made up of one noun and its *immediate* satellites: partitive, determiner, adjectives and adjective adverbs

a verbal sequence is made up of one verb (in all its forms: conjugated, infinitive, participle) and its *immediate* satellites: auxiliary, negation, non-subject clitics, verb adverb.

In both types of sequences, satellites depend on the central element: a noun or a verb.

If we remove nominal and verbal sequences from a sentence, we have left prepositions, subordination

or coordination conjunctions, relative pronouns, punctuation (commas, brackets, colons, full stops) and some adverbs. Let us define clips as these remaining segments: at the sequence hierarchic level, a clip is most often a single monosyllabic word (even a single comma); but it may contain up to 4 elements: comma - coordination conjunction - block adverb - preposition:

example: , and also from earlier work

At the sequence hierarchic level, a **sentence** is a *tripartition* of 3 types of contiguous segments: clips, nominal sequences and verbal sequences: a sentence may be completely coloured in 3 colours according to segment type. In the three corpora of scientific texts, the percentages of the 3 types of segments are in narrow forks: clip segments: 39% to 41%, nominal sequences: 40% to 47%, verbal sequences: 12% to 18%; the ratios of nominal sequences / verbal sequences are between 2/1 and 4/1.

two sentences segmented in sequences and clips:
(nominal sequences, **verbal sequences**, clips)

Many years ago such a project **might not have been undertaken** because it **was thought** that fish **emigrated** from their native stocks to perhaps a significant extent .

Dans les réseaux trophiques marins , de nombreuses molécules énergétiques **sont transférées** entre les différents niveaux d' organisation **structurant** les échanges .

II.1.b Blocks

block structure:

A block is made of a clip (on because) and a block body, which is made up of 1 to 3 sequences:

[on adult cod] [because it **was thought**]
[p N] [P N **V**]

At the block hierarchic level, clips mark the beginnings of blocks; they segment the sentence in blocks, they "clip", they attach every block into the sentence structure; they increase the ability to segment while listening or reading; they are few in number, and enable an unknown natural language to be easily segmented.

The central block of a sentence (central from the structural point of view) is the only block without a clip:

[Introduction] [N]
[such a project **was undertaken**] [N **V**]
[the Council **collects** information] [N **V** N]

The block body structures are:

central block (without a clip) / others (with a clip)

[N]	10% / **70%**	[p N]
[**V**] [**V** N]	0.5% / 20%	[c **V**] [q **I**]
[N **V**] [N **V** N]	**90%** / 10%	[P N **V**]
[**V** N]	0.5% /0.5%	[P **V** N]

(these statistics are given for the French corpus)

At the block hierarchic level, a **sentence** is made up of contiguous *or inserted* (about 1/5) blocks: some blocks surround other blocks.

The central block is generally the first block, but it may be preceded by some anteposed blocks.

two sentences segmented in contiguous blocks:
([blocks] are in square brackets)

[Many years ago]	[N p]
[such a project **was undertaken**]	[N **V**]
[because it **was thought**]	[P N **V**]
[that fish **emigrated**]	[P N **V**]
[from their native stocks]	[p N]
[to perhaps a significant extent]	[p w N]
[Dans les réseaux trophiques marins]	[p N]
[, des molécules **sont transférées**]	[, N **V**]
[entre les différents niveaux]	[p N]
[d' organisation]	[p N]
[**structurant** les échanges] .	[**R** N]

Blocks insertion is a major syntactic phenomenon, and its precise study clarifies problems of segmentation.

definition of block insertion:

the inserted block (indentation = 1) cuts the surrounding block into 2 non-empty parts (indentation = 0); the following examples are "topological block structures", exactly like the pretty printing of a block structured programming language:

[surrounding block: first part *indentation 0*
  [inserted block] *indentation 1*
 surrounding block: second part] *indentation 0*

[the International Council [N
  [for the Exploration] [p N]
  [of the Sea] [p N]
 collects information] **V** N]

insertion points of a block in another block:

These insertion points are few in number, because we observe that sequences are never cut.

We can deduce these 3 insertion points from the block structure:

[clip [3] subject [1] **verb** [2] object]

[1] between subject sequence and verb: 85%
[2] between verb and object sequence: 12%

[3] between clip and block body: 3%

some sentences segmented in contiguous blocks, with some inserted blocks, with indentation 1:

- [1] blocks inserted between subject and verb:

```
[ The exchange                          [ N
    [ between the Faroe Islands ]           [ p N ]
    [ and Faroe Bank ]                      [ c N ]
is perhaps intermediate ] .                 V ]

[ La mesure                             [ N
    [ de concentration ]                    [ p N ]
    [ de chlorophylle ]                     [ p N ]
 est utilisée ]                             V ]
[ pour  estimer la biomasse ] .         [ q I N ]
```

- [2] blocks inserted between verb and object:

```
[ on  attribue                          [ N V
    [ à  le point inconnu ]                 [ p N ]
 la classe ]                                N ]
[ de  son plus proche voisin ]          [ p N ]
```

insertion indentations of a block in another block:

Indentation 0 (non-inserted blocks) is the most frequent (76% to 82% in the three corpora); then comes the insertion with indentation 1, almost the only way to insert a block (24% to 18%); indentation 2 is very rare, almost only for prepositional blocks (<1%); indentation 3 has never been observed; these statistics are very regular for different natural languages.

On the contrary, segment insertion indentations of programming languages (e.g. ALGOL, Pascal, etc.) have no limit: that is why recursiveness is needed to describe their syntax.

inserted blocks typology:

The commonest inserted blocks are prepositional blocks (65%), followed by coordinated nominal sequences (16%), then past participles without an auxiliary (10%), then subordinated clauses: relative clauses, then circumstantial clauses (for French).

We notice that the more verbal a block is, the harder it is to insert it in another block.

a sentence segmented in blocks, with some inserted blocks, with indentation 1 and 2:

```
[ Les problèmes                         [N
    [ où  le nombre                         [pm  N
        [ de  les mesures ]                     [p N ]
      est important ]                       V ]
 induisent  un traitement ]             V  N ]
[ de  ce type ] .                       [p  N ]
```

II.2 Defining a linguistic relation between segments: dependency

II.2.a Determination dependency and actancial dependency

Tesnière places two types of dependency on the same level ([see Tesnière 59] pages 102 and 144): the dependency of an "actant" on a verb, which we can call "actancial" dependency, and the dependency of an adjective on a noun, which we can call "determination" dependency.

Let us focus on determination dependencies. For an adjective depending on a noun, a past participle on a nominal sequence, and a conjugated verb on its subject nominal sequence, agreement marks determination dependency (in French, German, Spanish, ...); examples:

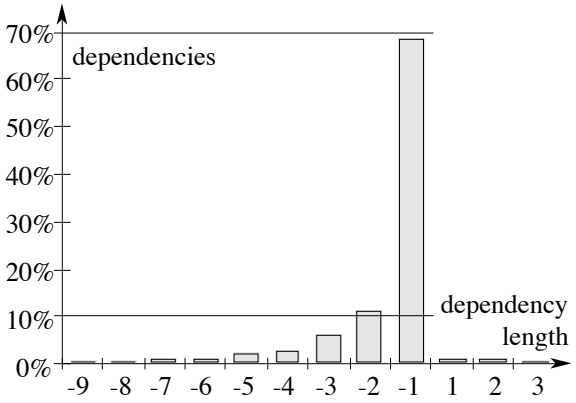
natural −−> mortality

```
[ the proportions  [recovered] were not shown ]
        •  <−−− -1−−−•
           <−−−−−−−−− -2−−−−−−−−•
```

II.2.b Determination dependency and contiguity at sequence level

If we observe *the same dependency type* (determination dependency) at *the same hierarchic level* (sequences), we have **homogeneous sight**: we then observe (for French) that most sequences (70%) depend on the preceding sequence (their "reigner", Tesnière's "régissant").

In other words, the most frequent mark of determination dependency between 2 sequences is **contiguity**. Let us define the dependency length as the number of sequences from a sequence to its "reigner" sequence: contiguity <=> dependency length = -1 (or 1, if anteposed). Here is the distribution of dependency lengths on the French corpus:



Which dependencies are not marked by contiguity?

A sequence does not depend on the preceding sequence only in the 4 following cases: the second part of a surrounding block, a coordinated sequence, an anteposed block, or a second postposed actant or circumstant of the same sequence.

In the sentence below (next page), every sequence depends on the preceding sequence (dependency length = -1), except two: **collects** depends on the Council 3 sequences before (dependency length = -3), and **make** depends on **collects** 4 sequences before (dependency length = -4).

an example of a dependency tree in which:

root, nodes and leaves are sequences: •

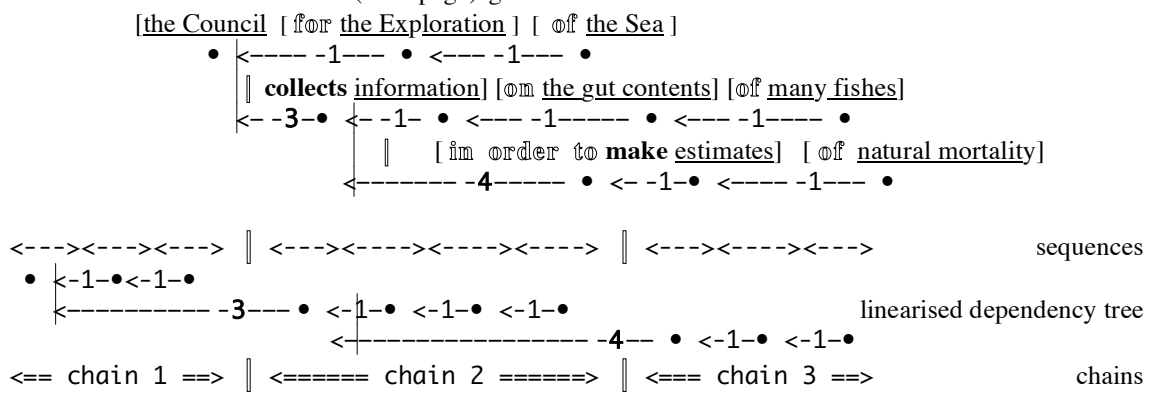
branches are determination dependencies between sequences: •<-- -1--• or •<-- -2--•

a node with one branch implies a dependency marked by contiguity: •<-- -1--•

a node with two branches implies a dependency marked by contiguity (the first branch) and another not marked by contiguity: the second branch continues (is linearised) when the first is finished (at its leaf): •<------- -2--------•

II.2.c A sequence dependency tree

Definition: the sentence below (next page) gives



II.3 Defining a segment made up of linked segments

II.3.a A segment made up of linked segments: the chain of contiguous dependent sequences

For the sentence above, we can see the shape of its dependency tree: this sentence is also partitioned into 3 segments in which every sequence depends on the preceding sequence. Let us call such a segment a **chain of contiguous dependent sequences**.

A cut (‖) between 2 chains occurs after the leaf of the first branch of a 2 branches node, where a sequence does not depend on the preceding one.

Regarding this cut (‖) between 2 chains, the longer dependencies are, the more frequent the comma is at the cut in writing, and the longer the pause in speech (see [López 93], chapter 4, pages 160 to 169). This means that the chain is probably a prosodic segment.

II.4.b Optimised linearisation of the dependency tree => chain partition of the sentence

When a node has more than one branch, the dependency tree is linearised into the one-dimensional sentence by sacrificing some sequence contiguities. This linearisation is done according to the following optimisation principle: a distended dependency is as short as possible. This implies that, when a node has more than one branch, the lightest branch (in number of nodes) is linearised first: branches are linearised in increasing weight order. This principle is weaker than some basic linguistic constraints. For instance, subject extensions remain contiguous to it, even when verb extensions are shorter.

III. Application to parsing of linear complexity in time

III.1 A definition of parsing

As said above (in I.4.b), automatic parsing will involve mainly 2 successive stages at every level:

1) delimiting and identifying segments of the written or spoken chain, at different hierarchic levels: words, sequences, blocks and sentences,

2) linking segments: words inside sequences, sequences inside blocks, and blocks inside sentences.

The parser outputs the only right analysis, because it should have all criteria to make all choices.

The parser must also be able to operate in a partially unknown environment, without having as data all possible written forms, all possible categories for a written form, and all possible sequence, block and sentence structures.

III.2 Linguistic properties of segments => segmentation strategies

III.2.a Defining word categories according to sequence partition

Word categorisation lies on theoretical bases: word categories are based on sentence tripartition in: words in nominal sequences, words in verbal sequences, and words in clip segments.

III.2.b Segment recognition while climbing the segment hierarchy

Segments are recognised while climbing the segment hierarchy, in stages of linear complexity:

1) recognising and pre-tagging words
2) recognising sequences,
   tagging and linking words in each sequence
3) recognising blocks,
   linking sequences in each block
4) building the dependency tree:
   linking sequences between different blocks

It should be noticed that only the most frequent sequence structures (in terms of word categories) and block structures (in terms of clip and sequences) are expected by the parser, but *sentence structures* (in terms of blocks) *are only observed*.

III.3 Main features of an algorithm of linear complexity in time

An important problem is to find an articulation between lexical and positional clues, and a chronology to use them.

III.3.a Pre-tagging words

1) dividing the sentence into words
amalgams are divided: *du -> de le*
elided words are separated: *l'eau -> l' eau*

2) giving a default pre-tag to every word, without an exhaustive dictionary

We can make a typology of 3 types of words, according to their neology type, with a specific solution for every type:

-a- grammatical words, with almost no neology, belong to small and finite sets: clips, determiners, pronouns, quantifiers, adverbs not derived from adjectives; solution: a small lexicon (400 forms in French), (words are pre-tagged at **55%** here); the output is a unique default category for every word, which will remain if no contextual deduction occurs on this word during the next stage. For example, *le la les* are determiners in most cases and sometimes object clitics: they are pre-tagged determiners;

-b- verbs, where neology is very low, belong to an almost finite set; solution: a lexicon of verb roots (about 7000 verbs in French ≈ 50 Kb) with codes for forms which are homographs of nouns or adjectives, and some ending rules to handle neology (words are pre-tagged at **17%** here);

-c- nouns, adjectives (and adverbs derived from adjectives), where neology is very rich, belong to an almost infinite set; solution (for alphabetic languages): extract ending rules from a base of forms (from BDLex); an ending rule enables possible categories (most often only one), genders, numbers to be deduced from an ending (about 500 rules are enough) (words are pre-tagged at **27%** here).

At this stage, if a word does not match up with any ending rule (**1%**), it is a noun or an adjective with an unknown gender and number.

III.3.b Recognising sequences => tagging and linking words in sequences

Sequence recognition is a symmetrical process for nominal sequences and verbal sequences; the strategy is to make the sequence tripartition appear, by colouring the words in 3 colours, spreading the colours inside sequences, and obtaining one colour and one category for every word:

1) colouring words in 3 colours by a propagation of local deductions

These deductions are done with interpreted declarative rules (< 100).

A local deduction rule has the form:

filter => new attributes

A rule concerns 2 to 5 contiguous words. For one word, the filter may concern the following attributes: colour (actually a generic category), category, written form, lemma, gender, number, person; Boolean operators may be used. For a given word, the new attributes may be a colour or a category.

They may be applied in two ways: the new attribute was either present or absent before application; this second way is mentioned by a prefixed attribute; it is used when there is no need to know anything about the word: the position in relation to the preceding words of the rule is enough to decide; this is used to tag a neologism, or to give a category, irrespective of the tag given at pre-tagging.

The propagation goes from left to right in the sentence: the chronology of understanding a sentence. For a given word, all rules are tested.

Their application is independent of their order: a rule compiler verifies that there is no pair of contradictory rules (non-empty filters intersection, and contradictory right members). Every time a rule is applied, sequence borders are updated.

Here are some totally sure linguistic principles on which the most frequently applied rules are built:

after a preposition (different from *d' de à pour*

sans après), a nominal sequence begins;
after a determiner, a nominal sequence continues;
after *je tu il on ils*, a verbal sequence begins;
between 2 prepositions, the first different from *d' de à pour sans après*, there is a noun.

2) studying every sequence: checking sequence structure and observing internal agreements

Nominal and verbal sequences structures are checked, and agreements are observed and computed: in nominal sequences, gender-number agreement between noun, determiner and adjectives, in verbal sequences with a form of "*être*", number agreement between "*être*" and attribute or past participle.

Inside sequences, dependencies on the central element (noun or verb) are computed.

3) if non-validated structure, finding a cut between 2 sequences

Two contiguous sequences of the same type have been coloured as a single sequence: the structure is then not validated, and the 2 sequences are separated into 2 parts, and their structures checked again.

4) updating the lexicon of the text

At the end of this step, sequences are recognised, words tagged and lemmatised, and the lexicon of the text updated. Sentences of the French corpus are correctly coloured at more than 99%, with a right word tagging better than 99%.

III.3.c Recognising blocks and linking sequences within each block

1) compressing sequences into a single code

Nominal sequences are coded: `N`

Verbal sequences are coded by type:

conjugated:	`V`	infinitive:	`I`
present participle:	`R`	past participle:	`ù`

All following processing is done on this pattern.

2) processing this pattern with interpreted declarative deduction rules (about 40)

These rules locate the beginning of blocks, recognise blocks or block parts made up of contiguous elements and compute dependencies within blocks, even if a block is cut in parts, when it surrounds other blocks. A deduction rule has the form:

filter => new attributes

A rule concerns 1 to 5 elements. For one element, the filter may concern the following attributes: sequence or clip category, block border ([]); Boolean operators may be used. For one element, the new attributes may be a block border and/or a relative integer which gives the relative position of the element on which it depends within the rule, even if the block is in 2 parts.

-a- locating beginnings of blocks (6 rules)

as blocks begin with clips, every clip marks the beginning of a block: `[ℙ` `[,` `[ℂ`

-b- recognising blocks or block parts (20 rules)

these segments are made up of contiguous elements: `[ℙ + N => 1 + ]`

-c- recognising 2 parts of the same block (15 rules)

The principle of this stage is to recognise the 2 edges of an insertion point (see insertion points in II.1.b above), even if there is no inserted block (contiguous edges): left edges are stacked (the only action of the right member of 6 rules): they are mainly subjects waiting for a verb, or verbs waiting for an object; right edges are recognised together with the nearest stacked left edge (9 rules): they are mainly verbs for a stacked subject, or objects for a stacked verb. Insertion indentation of inserted blocks is incremented; insertion indentation is a topological constraint to join 2 block parts of a same block: they must be at equal insertion indentation.

The expected result is obtained in more than 98% sequence dependencies *within blocks.*

III.3.d Building the dependency tree by linking sequences of different blocks

As said above (II.2.c) no pure positional means have been found today for linking 2 sequences of different blocks. Valuation functions have been used (see [Vergne 90]), and the expected result was obtained in about 85% dependencies *between different blocks*. But another hypothesis is now being explored: every sequence depends on the preceding sequence, at equal insertion indentation (a topological constraint), except in the 4 cases listed above (II.2.b). Some linguistic constraints are applied: for instance, a prepositional block without a determiner cannot be the "reigner" of a prepositional block with a determiner. The explored hypothesis is that topological and linguistic constraints are sufficient to link sequences *between different blocks*.

The first computed result is already of the same quality, but it will be improved, because we are in the early stages of exploring this hypothesis.

III.4 The linear complexity in time of the algorithm

This complexity may be evaluated in 2 ways: formal and practical.

On the formal point of view, the source program is made up of a constant number of one-level repetitive processes on the number of words or on the number of sequences of a sentence; practically, the cloud of points *parsing time – number of words of a sentence* shows the linearity of the phenomena (linear regression determination coefficient = 0.98).

Conclusion

This work attempts to renew the syntax of natural languages by conceptually separating segmentation from dependency, therefore defining (and observing on corpora) a hierarchy of non-recursive segments.

With these linguistic bases, it is possible to build non-combinatorial parsers, therefore working with linear complexity in time: "man triumphs over nature only by obeying it" (Francis Bacon, 1620).

References

[**López** 93] Eduardo **López Gonzalo**: *Estudio de técnicas de procedado lingüistico y acústico para sistemas de conversión texto-voz en español basados en concatenación de unidades* tesis doctoral, E.T.S.I. de Telecomunicación, Universidad Politécnica de Madrid, julio de 1993.

[**Lucas** 92] Nadine **Lucas**: *Syntaxe du paragraphe dans les textes scientifiques en japonais et en français* Colloque international: Parcours linguistiques de discours spécialisés, Université Paris III, septembre 1992, Peter Lang ed. 1993.

[**Lucas** 93] Nadine **Lucas**, **Nishina** Kikuko, **Akiba** Tomoyoshi, K.G. **Suresh**: *Discourse analysis of scientific textbooks in Japanese: a tool for producing automatic summaries* Departement of Computer Science, Tokyo Institute of Technology, March 1993.

[**Tesnière** 59] Lucien **Tesnière**: *Eléments de syntaxe structurale* Klincksieck (Paris) 1982.

[**Vergne** 90] Jacques **Vergne**: *A parser without a dictionary as a tool for research into French syntax* communication-demonstration at CoLing 90 International Conference on Computational Linguistics vol. 1 pp. 70-72, Helsinki, Finland, August 1990.

[**Vergne** 92] Jacques **Vergne**: *Syntax as clipping blocks: structures, algorithms and rules* communication and demonstration at SEPLN 92 congress (Sociedad Española para el Procesamiento del Lenguaje Natural), pp. 179-197 and 467, Granada, Spain, September 1992.

●