Encoding and reusing linguistic information expressed by Linguistic Properties
Caroline Hagège, Gabriel G. Bès

To cite this version:
Caroline Hagège, Gabriel G. Bès. Encoding and reusing linguistic information expressed by Linguistic Properties. COLING, Workshop on Grammar Engineering and Evaluation, 2002, Taiwan, Taiwan. <hal-01096754>
Encoding and reusing linguistic information expressed by Linguistic Properties

Caroline Hagège
Xerox Research Centre Europe (XRCE)
6 Chemin de Maupertuis
38240 Meylan - France
Caroline.Hagege@xrce.xerox.com

Gabriel G. Bès
GRILL, Université Blaise Pascal
34 Avenue Carnot
63000 Clermont Ferrand - France
Gabriel.Bes@univ-bpclermont.fr

Abstract

This paper presents a way to express linguistic knowledge independently of any algorithmic machinery and of any particular grammatical formalism. This is performed through Linguistic Properties, that will be presented. First, the status of linguistic knowledge in grammars is discussed, then the Linguistic Properties are presented and two experiments are mentioned. They illustrate the reusability of the linguistic information enclosed in these Properties.

1 Grammar and reusability of linguistic knowledge

One of the central points in linguistically-motivated natural language processing is the notion of grammar. It is commonly accepted that a grammar (see (TM90)) is intended, among other things, to be both a precise tool of natural languages’ description and the declarative data source which must be interpreted by a computer.

This means that the same metalanguage (the grammatical formalism) is used to encode the declarative linguistic informations and the rules that will feed a parser.

We consider that this double function of a grammar is a disadvantage from the point of view of the reusability of linguistic knowledge. Indeed, the same device (the grammar) contains both linguistic information and some information adapted to a specific goal (parsing (shallow or not), generation, etc.) and to a specific algorithmic machinery. In order to be reusable and suitable for different goals, we hold that linguistic knowledge must be free from any specific requirements, while being, nevertheless, formally expressed. Furthermore, it must be modularly organised within different levels of explicit granularity in order to offer a taylorisable access. Linguistic Properties, which we distinguished from Processes - i.e. effective computational procedures on strings of NL expressions - are a possible way to fulfill these requirements.

2 Linguistic Properties

Linguistic Properties (or simply Properties) were originally developed in the late 90’s within the 5P Paradigm. The 5 Ps stand for Protocole, systematic observations on sentences, Properties, linguistic declarative knowledge, Projections, generalizations on Properties of some natural language, Principles, cross-linguistic constraints on Projections or Principles and Processes that are effective computational procedures. We present in this section the kernel of Properties.

The following example introduces intuitively Properties and their potential of modularity.

Given the following French expression in (i), it is possible to distinguish, in its metalanguage, different layers or aspects, illustrated by (i-a) to (i-c).

\[
\begin{array}{l}
\text{(i)} \quad \text{les trois fleurs} \\
\text{(i-a) } \{ \text{les-[art, def], trois-[card], fleurs-[n]} \} \\
\text{(i-b) } \{ \text{les-[art, def]$_1$ troisi-[card]$_2$ fleurs-[n]$_3$}\}_N \}
\end{array}
\]

\[
\begin{array}{l}
\text{\{<1, 3>, <2, 3>, <3, 3>\}}
\end{array}
\]

(i) is a string of expressions; (i-a) is a set, each member of which is an expression of (i) associated to a category (or cat, see below), e.g. [art, def]; (i-b) is a parenthesised list obtained applying order relations to (i-a), and indexing it with the identifier $Nn$ (for Nominal nuclear phrase); (i-c) is obtained associating (i-b) to a set of pairs $<p, q>$, where $p$ and $q$ are positions in (i-b); the pair links the element in position $p$ to the element in position $q$.

We will say that (i-b) is a basic model reduced to its model string, namely (i-b), that (i-a) is a pack

For a complete and more formalised version, see (BH01).

The 5P Paradigm was presented as such in (Bé99) and (BH01) though antecedents, cited in (BH01), go back to a 199 report to the ESPRIT 393 ACORD european Project, where the notion of descriptive metalanguage, today’s Properties, were explicity introduced. For published work, see (BH05), (Hag98). For a different but related approach, see (BH05) and (Bla00) which remain “grammar oriented”.

Another layer, expressing a semantic representation, can be added, but it is not discussed in this paper.
associated to (i-b), that pairs \(<p, q>\) are *Arrowing pairs*, that (i-c) is an *arrowed model* incorporating the model string (i-b).

We distinguish three basic kinds of Properties: Existence Properties, Linearity Properties, Arrowing Properties. Packs as in (i-a) are specified by Existence Properties; order relations, by Linearity properties; Arrowing pairs, by Arrowing Properties.

Each identifier - e.g. \(Nn\) - has its associated set of Properties, e.g. Properties-\(Nn\), \(M-Nn\) being the set of all and only the models \(m-Nn\) satisfying Properties-\(Nn\). Properties are expressed on symbols which are \(cat\)'s or identifiers. From hereafter we use \(Sm\) as a metavariable on identifiers, and \(Sy\) as a metavariable on \(cat\)'s and \(Sm\)'s.

A cat is a set of label/value pairs, or, in reduced notation, a set of values (as in previous examples). A maximum categorie (\(mc\)) is a cat to which no other value can be added. The assumed Lexicon is a set of lexical entries, each one being an expression associated to one or more \(mc\)'s; \(cat\)'s subsume \(cat\)'s, if \(cat^i \subseteq cat^j\).

The whole system can be viewed as a modular axiomatic system in which models are the objects satisfying different kinds of Properties\(^3\). A basic model, as in (i-b), with its associated pack, as in (i-a), satisfies Existence and Linearity Properties; an arrowed model, as in (i-c) satisfies also Arrowing Properties. Furthermore, giving a set of Properties, a model can satisfy some, but not all of them. Properties can be expressed independently the ones from the others, and in any order. The set of features from which cat's are build can be more or less extended and, consequently, the granularity of cat's, and of Properties expressed on them, more or less refined.

The model substitution rule relates the identifiers \(Sm^1, \ldots, Sm^n\), each one with its associated M-\(Sm^i\). In a m-\(Sm^i\) with a \(Sm^k\), it substitutes some m-\(Sm^k\) for \(Sm^k\). E.g., assuming \((neg, adj)\)\(_2\)\(_{ADJ}\) as a French m-\(ADJ\) (underling, e.g. the string *pas belles*), the model substitution rule obtains (2) from the following (1).

1. \(<\text{art}_1 \text{ADJ}_n_2 \text{adj}_3 \text{n}_n\> \quad <1, 3>, <2, 3>, <3, 3> >$

2. \(<\text{art}_1 \text{neg}_2 \text{adj}_3 \text{n}_n\> \quad <1, 4>, <2, 3>, <3, 4>, <4, 4> >$

In an optimal situation, Properties associated to the \(Sm\)'s of some \(NL\), together with a Lexicon and the model substitution rule, specify the whole set of models required to describe the strings of expressions of the \(NL\). We concentrate in the following in the intuitive presentation of Properties specifying models obtained without the model substitution rule. Given the different kinds of Properties, we will intuitively characterise the conditions that must be fulfilled by a model in order to satisfy each one them \(^4\).

Subsumption is the basic relation linking models and Properties. We already defined above subsumption between \(cat\)'s. As a shorthand, we say here that \(Sm^1\) subsumes \(Sm^2\) if \(Sm^1 \subseteq Sm^2\). Furthermore, given sets \(S^1\) and \(S^2\) of \(Sy\) symbols, we say that \(S^1\) subsumes \(S^2\) if there is a bijective function between \(S^1\) and \(S^2\) such that each \(Sy^0\) in \(S^1\) subsumes its corresponding \(Sy^0\) in \(S^2\).

2.1 Existence Properties

Existence Properties associated to some M-\(Sm\) specify the set of packs from which any m-\(Sm\) is obtained. We distinguish five kinds: Vocabulary property, Unicity property, Nucleus Property, Exigency Property, Exclusion Property.

The Vocabulary Property, spelled by \(V_{Sm} = \{Sy^1, \ldots, Sy^n\}\) says that each symbol in the pack associated to a \(m-\Sm\) is subsumed by some symbol in \(V_{Sm}\), and each symbol in \(V_{Sm}\) subsumes some symbol in the pack of some \(m-\Sm\). E.g. (singleton categories are spelled with their value) French \(V_{NN} = \{\text{det, poss, card, noun}..\}\) is the vocabulary for \(Nn\) (nominal nuclear) French phrases (roughly, nominal chunks), assuming \(mc\)'s: \{\text{det, art, def...}, \text{det, art, ind...}, \text{det, dem...}\}, which are associated in the Lexicon to, respectively, the expressions \{\text{les, la, le...}, \text{un, une, des...}\}, \{\text{ce, ces, cette...}\}.

The Unicity Property, spelled by \(U_{n_{Sm}} = \{Sy^1, \ldots, Sy^n\}\) says that there are no two symbols in the pack associated to a \(m-\Sm\) subsumed by one and the same symbol in \(U_{n_{Sm}}\). E.g. French \(U_{NN} = \{\text{det, card...}\}\) express that there are no two articles, or two demonstratives or an article and a demonstrative in a \(Nn\) phrase.

The Nucleus Property, spelled by \(N_{n_{Sm}} = \{Sy^1, \ldots, Sy^n\}\) says that in each \(m-\Sm\) there is one and only one position with a nucleus symbol - spelled \(Sy\) - subsumed by some symbol in \(N_{n_{Sm}}\). E.g. French \(N_{n_{Sm}} = \{\text{card, quant, noun}..\}\) express that \(Nn\) phrases can have as a Nucleus either a cardinal (e.g. \(il\) a \(vu (\text{les trois})_Nn\)) or a quantifier (e.g. \(il\) a \(vu (\text{les} \text{ tous})_Nn\)), or a noun (e.g. \(il\) a \(vu (\text{les} \text{ fleurs})_Nn\)).

The Exigency Property, spelled by \(S^0 \rightarrow_{Sm} \{S^1, \ldots, S^n\}\)

\(^3\)The system benefits from the concept of factorizing relations of standard production rules of now traditional grammars. See in particular the LP statements of GPSSG dissociated from dominance ID rule, and dependency grammar ([Tes00]), early HPSG in ([PAS87]). The system tries to push this basic idea to its limits, dissolving thus the concept of production rules. An analog of what the system of Properties is expected to express compared to production rules, can be seen in regular expressions as compared to production grammars of type 3 in the Chomsky’s hierarchy.

\(^4\)For a more formal and complete presentation, see BH-01.
saying (remember that S’s spell sets of Sy) that if in
the pack of a m - Sm there is included a set of
symbols S^k subsumed by S^0 there must be also some
S^r included in the pack such that S^r is subsumed by
S^{r+1}. E.g. French \{[n, c] \rightarrow Nn \{[det], [card], \ldots \},
where [n, c] stands for common nouns, express that
common nouns require a determiner or a cardinal.
The Exclusion Property, spelled by
S^0 \not\preceq_S S^k \ldots \not\preceq_S S^p.
It says that if in a m - Sm there is a symbol subsumed by
S^k and a symbol subsumed by S^{k+1}, the for-
ter precedes the latter.
E.g. : in French m-Nn’s, a quantifier tout(e,s) pre-
ces all other cat’s, which is expressed by quant
\not\preceq_Nn n, det, poss, card...

2.3 Arrowing Properties
The basic role of Arrowing Properties is to specify the
graph - i.e. the set of Arrowing pairs - that is the
backbone from which the semantic representation is build. An arrowing pair (Ar) is a pair \{p, q\},
where p and q are positions in the model string, and
which can be understood as ”the Sy in position p
arrows to the Sy in position q”. An Ar is thus an arc
between two Sy’s. Ar’s are expressed by arrowing
formulae, which, in their simplest form, are spelled
Sy^p \rightarrow_S S^q. It is also possible to spell dis-
junctive arrowing, expressing that some Sy arrows
to either Sy^p or to Sy^q. By a general convention,
a nucleus Sy^p arrows to himself. General conditions
limit the expressive power of Arrowing formulae, as-
suring, among others, that the resulting graph must
be connected, and, with the exception of the reflexive
arrowing of Sy, acyclic.
E.g., among French Arrowing formulae, there is
quant \not\rightarrow_Nn Sy^p, where Sy^p is a variable
on the Nucleus and which express that the quanti-
fier tous arrows to any Nucleus in a Nn phrase:
\langle(tous_1 \not\preceq_gnrois_2)_Nn, \langle 1, 2 >, \langle 2, 2 \rangle >, \langle(tous_1
les_2 \not\preceq_garons_3)_Nn, \langle 1, 3 >, \langle 2, 3 \rangle, \langle 3, 3 \rangle \rangle.

3 Exploring properties
Two experiments have been carried out in the ex-
ploration of Existence and Linearity properties. In
the first experiment, Linguistic Properties were used
to derive the linguistic data structures used by a
chunker and a NP extractor for Portuguese (see
(BHC99)). In the second experiment, Linguistic

Properties were used to structure lexical entries in
an HPSG-style grammar (see (HB98) and (Hag00)).
In both cases, the basic idea is the same: associate
to each category declared for a given model the com-
binatorial information attached to this category in a
certain grammatical context.
We describe here these two experiments in more de-
dtails

3.1 First experiment
3.1.1 Context
A fine grained description of the Portuguese NP has
been accomplished with Linguistic Properties and
we wanted to use this linguistic description in or-
der to extract NPs from Portuguese running texts.
In a first step, the input text is tokenized and mor-
phologically analyzed (SMORPH (AM98)). Then,
the tokenized and morphologically analyzed text is
processed, eliminating partially some ambiguity
and grouping or ungrouping some tokens previously
delimited (MPS). Then the text is chunked and fi-
ally, NPs (defined as regular expressions of chunks)
are extracted. Figure 1 summarizes the processing
chain for NP extraction.

Our chunker (called AF) consists in a very simple
algorithm (see (BHC99)) which uses linguistic struc-
tures (called leaves) associated to each token of
the text and tries to concatenate these structures from
left to right until the end of the text. Each concate-
nation introduces constraints for the next concate-
nation and, during parsing, part of the ambiguity is
solved as a side effect when concatenation fails.
To illustrate intuitively how our chunker works, as-
sume we want to analyze the following string with
the following leaves.

As danças
(The dances)

Leaf 1 This leaf is associated to As
• The lemma associated to As is o
• The category is a definite article
• The model where this category appears is nom-
inal chunk
3.1.2 Leaves and Leaf Patterns
A leaf is thus a structure of the following form (We represent it as a Prolog predicate).
leaf(WF, L, Cat, ModId, BStat, EStat, Foll). Where:

- WF (Word Form) is the token found in the text to analyze
- L (Lemma) is the corresponding lemma
- Cat (category) is the corresponding category
- ModId (Model Identifier) identifies the model in which this category can appear
- BStat (Begin Status) is the integer 0, 1 or 2 meaning respectively that this category never, always or sometimes starts the model identified by ModId
- EStat (End Status) is the integer 0, 1 or 2 meaning respectively that this category never, always or sometimes ends the model identified by ModId
- Foll (Followings) is the set of categories that can follow the category Cat in the model identified by ModId (The empty set when EndStatus is 1)

We call a Leaf Pattern a leaf structure in which the first argument (the word form) is not instantiated. Our problem here is to deduce, from the Properties, all the Leaves Patterns that are necessary to analyze one text.

3.1.3 Relations between categories appearing in a given model string
Given the vocabulary V of some model identifier Sm, it is possible, using Existence Properties and Linearity Properties to define the following relations in V x V⁶.

precede:  a precede b if in any m – Sm containing a and b, a always precedes b

order:  a order b if there is at least a m – Sm in which it is possible to say that a precedes b or that b precedes a.

exige:  a exigere b if each m – Sm where a appears, b also appears.

exclu:  a exclu b if there is no m – Sm with a and b.

It is also possible to define two subsets of V, S₀ and S₁. So consists of the elements of V that are always alone in a model string and is defined the following way:

⁶In the following section, we make two simplifications: the notion of sumbsumption between categories in no taken into account and we do not consider models within models, but the general idea keeps the same
\[ So = \{ a \in V \mid \forall b \in V \text{ exclu}(a, b) \} \]

\( S1 \) is the complementary of \( So \) in \( V \)

For each category \( a \) of \( V \), it is also possible to define the set \( LP_a \) as the set of all categories that possibly follow \( a \) in at least one model string.

Having these relations and these sets, one can define the subsets of \( V \) that always, sometimes and never start a model string and the subsets of \( V \) that always, sometimes and never end a model string, which is precisely what is needed to define the leaves together with \( LP_a \).

We called these subsets \( AS \) (Always start), \( SS \) (Sometimes start), \( NS \) (Never start), \( AE \) (Always end), \( SE \) (Sometimes end) and \( NE \) (Never end)

With these definitions and considering the set of Properties that define the models identified by \( m \)-\( Sm \) we can then construct a set of leaf patterns the following way:

- The first argument is a variable (that will be then instantiate with a linguistic form present in the text)
- The second argument of the leaf predicate is instantiated to an element of \( V \)
- The third argument of the leaf predicate is instantiated to \( m \)-\( Sm \).
- The fourth argument of the leaf predicate is instantiated to \( 1, 2, 0 \) according to the fact that this element is member of \( AS, SS \) or \( NS \).
- The fifth argument of the leaf predicate is instantiated to \( 1, 2 \) or \( 0 \) according to the fact that this element is member of \( AE, SE \) of \( NE \).
- The sixth argument corresponds to the set \( LP_{cat} \), being \( cat \) the category that is present in the second argument.

### 3.2 Second experiment

In this second experiment, we want to use the Properties defined for the nominal chunk in order to construct lexical entries that can enable to analyze nominal chunks in an HPSG-style (see (CS94) and (SW99)). The HPSG grammar was then implemented in ALE (Attribute Logic Engine, developed by B. Carpenter and G. Penn). Only the syntactic part of the lexical entries is taken into account.

We decided that for our grammar a nominal chunk has to be a saturated sign with a nominal head. The analysis fails if:

- No analysis is produced
- A linguistic sign is obtained but it is not saturated

#### 3.2.1 What we have to consider

We have to take into account the structuration of linguistic signs that HPSG formalism stipulates. That is:

**In the type hierarchy** A linguistic sign has in the path SYN:SYN:LOC:CAT:HEAD (from now on the whole path is designed by HEAD) a value of type head that has the following subtypes.

\[ \text{head} \]

\[ \text{func} \]

\[ \text{det} \]

\[ \text{mark} \]

**In the structuration of lexical signs** If the value of HEAD is noun then there is a value for the path SYN:SYN:LOC:CAT:VAL:SPR (from now on just VAL:SPR) which is of type list of linguistic signs

If the value of HEAD is det then the value of the path SYN:SYN:LOC:CAT:HEAD:SPEC (from now on just SPEC) is of type non-empty list of linguistic signs.

Finally, if the value of HEAD is \( adj \) then the value of VAL:SPR is the empty list and the value of SPEC is the empty list.

### 3.2.2 What we can infer from Linguistic Properties

**Definition of the set of categories that never can be alone in a nominal chunk model** Considering the set of Properties modelling nominal chunks, we can define the subset \( S2 \) of the vocabulary \( V \) consisting in the set of categories that never can be alone in a model.

\[ S2 = \{ a \in V \mid \exists b \in V \text{ exige}(a, b) \} \]

**Rule 1** All the categories that are members of the above defined sets \( AE \cup SE \) must have the value noun for HEAD. Nouns and nominalized adjectives that can be the head of a nominal chunks are concerned by this rule.

**Rule 2** All the categories that are member of the set \( So \) (defined above) must be associated to a lexical entry with the value empty list for VAL:SPR. Plural nouns and pronouns that can be used alone in a nominal chunk are concerned by this rule. Note that Rule 2 applies to all the categories for which Rule 1 applies too as \( So \) is included in \( AE \).
Rule 3  All the categories that are members of $\text{AS} \cup \text{SS} \cap S_2$ and that are not considered traditionally as adjectives have the value *det* for HEAD and have for SPEC a value of type *sign* that is subsumed by SYNSEM:SYN:LOC:CAT:HEAD:noun. Determiners are concerned with this rule.

Rule 4  This rule handles with possible combination of determiners (or determiners and quantifiers) and gives one possibility to combine them together. It stipulates that if a category treated in Rule 3 can precede another category treated in Rule $3^7$ (we know that through the relation *order* defined above), then it is necessary to provide either a complex determiner structure, or to add to the VAL:SPR value of all the categories treated in Rule 1 the whole list of determiners.

Rule 5  Any category of $S_2$ that has not be considered by Rule 3 are taken as adjective and have the value *adj* for HEAD.

3.3 Extensions

It is well known that there are different kinds and different sources of ambiguity. We point here two of them and how they can be treated within our framework.

A linguistic expression can be associated in the Lexicon to more than one *mc* : it is, e.g., the case for Leaf 2 and 3 of the first experiment in Section 3.1. The ambiguity is there resolved thanks to Leave 1. Suppose that, as in French, there is a string of expressions in a related pattern - as *le juge* - where both expressions are ambiguous (*le* being an article and a clitic, *juge* a noun or a verb). In this situation, the ambiguity is maintained, the system specifying both *m-Nn* and *m-Vn* for the *le juge* (respectively, a nominal and a verbal chunk). This ambiguity will be resolved in a context - e.g. to the right of a preposition *Leaf* - in which the expression can follow if it is specified as *m-Nn* but not if it is as *m-Vn*.

As an important side-effect of the first experiment (Section 3.1), it is remarked in H00 (these) that applying the processing chain (see Figure 1) to previously and independently disambiguated expressions improves very little the final results. We think that observations as this one indicate that the incremental tactic of bottom-up parsing and that the requirement of a disambiguation layer before parsing is not the only possible way.

In the experiments presented in this paper we work on model strings build with *cat's*, not with *Sm* symbols (identifiers). Two basic types of identifiers are recognized: the one related to nuclear phrases or chunks, which are spelled *Xn*. *X* being a variable on *N*, *V*, *ADJ*...and the ones related to not nuclear phrase, spelled with the bare *X* and its possible instantiations. In general, a *X'n* in the model string of some *X'n* are not ambiguously related. But attachment of *X'n* to the right of a pattern *X'n*...*X'n* can be ambiguous.

Properties here presented apply exactly the same on models string with or without *Sm's*. So the previously characterised ambiguity can be expressed by disjunctive arrowing in arrowing formula (see Section 2.3).

4 Conclusion

In current work on syntax (heuristics for robust parsing (see (AMCR01), (TJ97)) or unification-based grammatical formalisms), it is quite difficult to access pure linguistic information since the same syntax is used both for the linguistic description and the rules for the parsers. We believe that the expression of linguistic information by means of Linguistic Properties is a possible step in the direction of the centralization of linguistic knowledge with the following benefits:

- Syntacticians would spend less time rewriting rules carrying the same information for different formalisms or for different parsers.
- The construction of a grammatical reference, expressed in a formalized and non-ambiguous way.

The notion of a grammar as a source of linguistic knowledge is thus revisited in favor of a notion of linguistic knowledge base8 from which syntactic information could be extracted for one or another specific grammar or application. The two experiments that we described above seem to be a step in this direction.

References


---

7 In Portuguese, combination of determiners and quantifiers.

8 The idea of a linguistic knowledge base was originally mentioned by G. G. Bès in a project proposal (Cale) submitted in 1991.


