From a Surface Analysis to a Dependency Structure

Luísa Coheur
L^{2}F INESC-ID / GRIL
Lisboa, Portugal
Luisa.Coheur@l2f.inesc-id.pt

Nuno Mamede
L^{2}F INESC-ID / IST
Lisboa, Portugal
Nuno.Mamede@inesc-id.pt

Gabriel G. Bès
GRIL / Univ. Blaise-Pascal
Clermont-Ferrand, France
BesGabriel@yahoo.fr

Abstract

This paper describes how we use the arrows properties from the 5P Paradigm to generate a dependency structure from a surface analysis. Besides the arrows properties, two modules, Algas and Ogre, are presented. Moreover, we show how we express linguistic descriptions away from parsing decisions.

1 Introduction

Following the 5P Paradigm (Bès, 1999; Hagège, 2000; Bès and Hagège, 2001) we build a syntactic-semantic interface which obtains a graph from the analysis of input text. The graph express a dependency structure, which is the domain of a function that will obtain as output a logic semantic interpretation.

The whole syntactic-semantic interface is integrated by four modules: Susana in charge of surface analysis, Algas and Ogre, defining the graph, and ASdeCopas, that obtains the logic semantic representation. In this paper we present the first three modules, focussing mainly on Algas and Ogre.

5P argues for a carefully separation between linguistic descriptions and algorithms. The first ones are expressed by Properties and the last ones by Processes. Futhermore, linguistic modelised and formalised descriptions (i.e. Properties, P2 of 5P) are not designed to be the declarative source of algorithms, but rather as a repository of information (Hagège and Bès, 2002) that one should be able to re-use (totally or partially) in each task. Following and completing this, we assume that the parsing issue can be viewed from at least three different points of view: (i) modelised and formalised linguistic observation; (ii) computational effective procedures; (iii) useful computational constraints. These three aspects of the same issue are distinctly tackled in the proposed syntactic-semantic interface, but they converge in the obtention of results.

There are three different kinds of Properties (P2) in 5P: existence, linearity and arrow properties. The first two underly the Susana module (3.1). They express which are the possible morphological categories of some expression and the possible order between them. The third ones arrow properties specify arrow pairs, which formally are directed arcs of a graph. Arrow properties underly the Algas (3.2) and Ogre (3.3) modules. At the level of Projections (i.e. P3 of 5P) the balanced parentheses structure underlying sentences is exploited (2). Computational useful constraints improve Algas performance (5).

2 Arrow properties

The motivation behind an arrow property is to connect two elements, because the established relation is needed to reach the desired semantic representation (Bès, 1999). Notice that this formalism can be applied to establish dependencies either between words, chunks or phrases. Nevertheless, arrows can be seen as dependencies but, contrary to the main dependency theories, an arrow is not labeled and go from dependents to the head (Hagège, 2000).

Let C be the set of category labels available, M the set of chunk labels, P a set of phrase labels and I a set of indexes.

Arrow Property: An arrow property is a tuple (X, n, Z, Y, m, R^+, R^-) noted by:
$\mathsf{X}_n \rightarrow_{\mathsf{Z}} \mathsf{Y}_m$,
$+\mathsf{R}^+$
-R^-
where:

- X, $\mathsf{Y} \in \mathsf{M} \cup \mathsf{C}$ (X is said to be the source and Y the target of the arrow);
- $\mathsf{Z} \in \mathsf{M} \cup \mathsf{P}$ (the segment labeled Z contains X and Y);
- R^+, R^- are sets of constraints over the arrows (respectively, the set of constraints

that Z must verify, either positive ones (R^+) on symbols which must be attested or negative ones (R^-) on symbols which must not occur);

- n, m $\in$ I.

Both R^+, R^- impose simple constraints over the arrows, such as symbols that should or should not occur within Z or linear order relations that should be satisfied between its constituents. As an example, the following arrow property says that within an interrogative phrase (Pint), an interrogative chunk (IntC) with an interrogative pronoun inside (pint) arrows a nominal chunk (NC) on its right ($i \prec k$), as long as there is no other nominal chunk between them ($i \prec j \prec k$).

$\mathsf{IntC}_i(\{\mathsf{pint}\}/) \rightarrow_{Pint} \mathsf{NC}_k$
$-\{\mathsf{NC}_j\}$

A more complex type of constraint is the "stack" constraint (Coheur, 2004). This constraint is based on the linguistically motivated work over balanced parentheses of (Bès and Dahl, 2003; Bès et al., 2003). Briefly, the idea behind that work is the following: given a sentence, if we introduce a left parentheses everytime we find a word such as *que*(*that*), *se*(*if*), ...) – the *introducers* – and a right parentheses everytime we find an inflected verbal form[1], at the end of the sentence, the number of left parentheses is equal to the number of right ones, and at any point of it, the number of left ones is equal or greater that the number of right ones (Bès and Dahl, 2003). In (Bès and Dahl, 2003), they use this natural language evidence in order to identify the main phrase, relatives, coordinations, etc. Within our work, we use it to precise arrowing relations. For example, consider the sentence *Quais os hotéis que têm piscina?*(*Which are the hotels that have a swimming pool?*). The surface analysis of this statement results in the following (where VC stands for verbal chunk):

$(Quais)_{IntC}$ $(os\ hotéis)_{NC}$ $(que)_{RelC}$ $(têm)_{VC}$ $(piscina)_{NC}$

Typically the NC *os hotéis* arrows the main VC, but in this situation, as there is no main VC we want it to arrow itself. Nevertheless, there is an arrow property saying that an NC can arrow a VC, which applied to this particular situation would establish a wrong dependency (Figure 1).

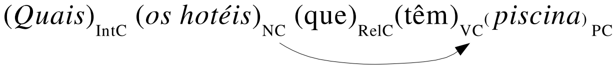

Figure 1: Wrong dependency

Roughly, we use the stack constraint that says that an NC arrows a VC if the stack of introducers and flexioned verbs is empty between them[2]:

$\mathsf{NC}_i \rightarrow_S \mathsf{VC}_k$
$+\{\mathrm{stack}_j = [\,]\}$

As a result, if we consider again the example *Quais os hotéis que têm piscina*, the NC *hotéis* will not arrow the VC *têm*, because the stack constraint is not verified between them (there is only the introducer *que*).

3 Reaching the dependency structure

3.1 Surface analysis

From existence and linearity properties (P2 of 5P) specifiyng chunks, it can be deduced what categories can or must start a chunk, and which ones can or must be the last one. Drawing on this linguistic information, chunks are detected in a surface analysis made by Susana (Batista and Mamede, 2002). As an example, consider the question *Qual a maior praia do Algarve?*(*Which is the biggest beach in Algarve?*). Susana outputs the following surface analysis (where PC stands for prepositional chunk):

$(Qual)_{IntC}$ $(a\ maior\ praia)_{NC}$ $(do\ Algarve)_{PC}$ $(?)_{Ponct}$

3.2 Algas

Algas is the C++ program responsible for connecting chunks and the elements inside them, taking as input a structure that contains information from arrow properties and also information that can limit the search space (see section 4 from details about this). Additionally, as inside the majority of the chunks all the elements arrow the last element (the head), the user can declare which are the chunks that verify this property. As a result, no calculus need to be made in order to compute dependencies inside these chunks: all its elements arrow the last one. This possibility is computational very usefull.

[1] See (Bès and Dahl, 2003) for details about how to deal with coordination.

[2] In fact, this restriction is a little more complicated than this.

Continuing with our example, after Algas execution, we have the output from Figure 2. Both the IntC and the PC chunks arrow the NC and inside them, all the elements arrow the head.

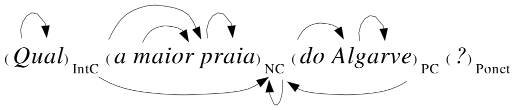

Figure 2: Algas's output.

Algas is able to skip unalyzable parts of a sentence, but (for the moment) some constraints are made to its output:

(1) There is at most an element arrowing itself, inside each chunk;

(2) Cycles are not allowed;

(3) Arrow crossing is not allowed (projectivity);

(4) An element cannot be the target of an arrow if it is not the source of any arrow.

Notice that these constraints are made inside the program. Notice that, in particular the projectivity requirement is not imposed by 5P. We impose it, due to the fact that – for the moment – we are only dealing with written Portuguese, that typically respects this property.

3.3 Ogre

After Algas, the text is processed by Ogre, a pipeline of Perl and XSLT scripts, that generates a graph from the arrowed structures produced by Algas[3]. This process is based on the following: if a chunk arrows another chunk, the head of the first chunk will arrow the head of the second chunk, and the chunk label can be omitted.

Continuing with our example, after Ogre we have the graph of Figure 3 (a dependency structure). Basically, IntC and PC head – respectively *qual* and *Algarve* – arrow now the NC head.

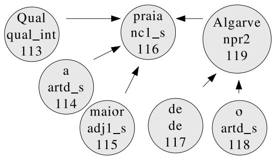

Figure 3: Ogre's output.

It might seem that we are keeping away information in this step, but the new arrowing relation between chunk heads keeps the lost structures. Beside, as information about the direction of the arrows is kept, and the position of each word is also kept in the graph, we are still able to distinguish behaviours dependent on word order for the following semantic task. That is, both semantic relations and word order are kept within our graph.

Ogre's motivation is to converge different structures into the same graph. For example, after Ogre's execution *O Ritz é onde?*, *É onde o Ritz?* and *Onde é o Ritz?*, they all share the same graph (appart from positions).

4 From descriptions to the algorithm input structures

In order to keep descriptions apart from processing, arrow properties and Algas input structures are developed in parallel. Then, arrow properties are formally mapped into Algas input structures (see (Coheur, 2004) for details). This decision allowed us to add computational constraints to Algas input structures, leaving descriptions untouchable.

In fact, in order to reduce the search space, Algas has the option of letting the user control the distance between the source and the target of an arrow. This is particularly very usefull to control PP attachments (in this case PC attachments). Thus, if we want a PC to arrow an NC that is at most n positions away, we simply say:

$$\mathsf{PC} \rightarrow_S \mathsf{NC}\ [\{\mathsf{NC} <_n \mathsf{PC}\}/]$$

Notice that we could make an extension over the arrow properties formalism in order to allow this kind of information. Nevertheless, it is well know that in natural language there is no fix distance between two elements. Adding a distance constraint over arrow properties would add procedural information to a repository resulting from natural language observations.

5 Applications

Both Algas and Ogre are part of a syntactic-semantic interface, where the module responsible for the generation of logical forms is called AsdeCopas (Coheur et al., 2003). This interface has been applied in a semantic disambiguation task of a set of quantifiers and also in question interpretation.

[3] Arrowed structures produced by Algas can also be seen as a graph, having nodes containing graphs.

Notice that, although arrows are not labeled, the fact that we know its source, target and direction, give us enough information to find (or at least guess) a label for it. In fact, we could add a label to the majority of the arrows. For example, using the link-types from the Link Grammar (Sleator and Temperley, 1993; Sleator, 1998), if an adverb connects an adjective, this connection would be labeled **EA**, if an adverb connects another adverb, the label would be **EE**. AsdeCopas can be used to add this information to the graph. Nevertheless, the fact that we are using an unlabelled connection serves languages as Portuguese particularly well. In Portuguese, it is not 100% sure that we are able to identify the subject. For example, we can say "*O Tomás come a sopa.*", "*Come a sopa o Tomás.*", or even "*A sopa come o Tomás.*" having all the same (most probable) interpretation: *Thomas eats the soup*. That is, there is no misleading interpretation due to our knowledge of the world: a man can eat a soup, but a soup cannot eat a man. As so, arrow properties simply establish relations, and we leave to semantic analysis the task of deciding what is the nature of these relations.

6 Conclusions

We presented two modules – Algas and Ogre – that build a dependency graph from a surface analysis. Algas uses information from a formalism called arrows properties. Nevertheless this formalism is independent from Algas input structures, that can be enriched with information that limits the relations to establish.

In the future we want the user to be able to control the constraints over Algas output. That is, the user will have the option to chose if output may contain arrows crossing or not.

For the moment the Susana-Algas-Ogre modules of the syntactic-semantic interface behave without problems in the domain of question interpretation. They apply successfully to an elicited corpus of questions produced by N portuguese speakers which were asked to produce them simulating effective and natural questions. Our next step is to try to use them incrementally (Aït-Mokhtar et al., 2002).

Also, another improvement will be over arrow properties, as we want to organise them in a hierarchy.

7 Acknowledgements

This paper was supported by FCT (Fundação para a Ciência e Tecnologia) and by Project POSI/PLP/41319/2001 (FEDER).

References

Salah Aït-Mokhtar, Jean-Pierre Chanod, and Claude Roux. 2002. Robustness beyound shallowness: incremental deep parsing. *Natural Language Engineering*, pages 121–144.

Fernando Batista and Nuno Mamede. 2002. SuSAna: Módulo multifuncional da análise sintáctica de superfície. In Julio Gonzalo, Anselmo Peñas, and Antonio Ferrández, editors, *Proc. Multilingual Information Access and Natural Language Processing Workshop (IBERAMIA 2002)*, pages 29–37, Sevilla, Spain, November.

Gabriel G. Bès and Veronica Dahl. 2003. Balanced parentheses in nl texts: a useful cue in the syntax/semantics interface. In *Nacy Workshop on Prospects and Advances in the Syntax/Semantics Interface.*

Gabriel G. Bès and Caroline Hagège. 2001. Properties in 5P. Technical report, GRIL, Université Blaise-Pascal, Clermont-Ferrand, France, November.

Gabriel G. Bès, Veronica Dahl, Daniel Guillot, Lionel Lamadon, Ioana Milutinovici, and Joana Paulo. 2003. A parsing system for balanced parentheses in nl texts. In *CLIN'2003.*

Gabriel G. Bès. 1999. La phrase verbal noyau en français. In *in Recherches sur le français parlé*, volume 15, pages 273–358. Université de Provence, France.

Luísa Coheur, Nuno Mamede, and Gabriel G. Bés. 2003. ASdeCopas: a syntactic-semantic interface. In *Epia*, Beja, Portugal, Dezembro. Springer-Verlag.

Luísa Coheur. 2004. *A interface entre a sintaxe e a semântica no quadro das línguas naturais.* Ph.D. thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal, Université Blaise-Pascal, France. work in progress.

Caroline Hagège and Gabriel G. Bès. 2002. Encoding and reusing linguistic information expressed by linguistic properties. In *Proceedings of COLING'2002, Taipei.*

Caroline Hagège. 2000. *Analyse Syntatic Automatique du Portugais.* Ph.D. thesis, Université Blaise Pascal, Clermont-Ferrand, France.

Daniel Sleator and Davy Temperley. 1993. Parsing english with a link grammar. In *Proceedings of the Third International Workshop on Parsing Technologies*.

Daniel Sleator. 1998. Summary of link types.