The Meaning of Syntactic Dependencies*

Pablo Gamallo Otero (Santiago de Compostela)

Abstract

This paper discusses the semantic content of syntactic dependencies. We assume that syntactic dependencies play a central role in the process of semantic interpretation. They are defined as selective functions on word denotations. Among their properties, special attention will be paid to their ability to make interpretation co-compositional and incremental. To describe the semantic properties of dependencies, the paper will be focused on two particular linguistic tasks: word sense disambiguation and attachment resolution. The second task will be performed using a strategy based on automatic acquisition from corpora.

1   Introduction

The main proposal of this paper is to sketch a semantic interpretation of syntactic dependencies. A syntactic dependency will be semantically defined as a binary operation that takes as arguments the denotations of the two related words (both the *head* and the *dependent*), and gives as result a more elaborate arrangement of their denotations. It is assumed here that such a binary operation plays an essential role in the process of semantic interpretation. In particular, this paper will analyse two disambiguation tasks in which dependencies are involved: word sense disambiguation and structural ambiguity resolution. The second task will be performed using a unsupervised method based on automatic acquisition of selectional restrictions.

Another contribution of this article is to discuss some issues on the computational and psycholinguistic motivation of the semantic description of dependencies. This discussion will not be exhaustive; it will only serve to check if our semantic model is or is not useful for different linguistic tasks and applications.

Dependencies have been traditionally considered as syntactic objects. They are at the centre of many syntactic theories, known as dependency-based approaches: e.g., Dependency Grammar (cf. Tesnière 1959), the Mel'čuk's Meaning-Text Theory (cf. Kahane 2003), or Word Grammar (cf. Hudson 2003). In these theories, the main properties of syntactic dependencies are the following. First, they are relations between individual words. It follows that words and dependencies are the only units of Syntax. Syntactic analysis no longer uses phrases to describe sentence structure, because everything that needs to be said can be said in terms of dependencies between single words. Second, they are asymmetrical relations (except coordination). One word is always subordinated (dependent) to the other, called the head. In Figure 1, asymmetry is noted using an oriented arrow, which goes from the dependent word, "Mary", to the head word, "ran". Third, dependencies are individualised by labelled links imposing some linguistic conditions on the linked words. Figure 2 illustrates a *subj* dependency (subject) without specific words assigned to the head and dependent roles.

According to the morpho-syntactic conditions imposed by this dependency, we know that the head, *x*, must be a verb while the dependent, *y*, is a noun (or a nominalised expression). More elaborated labels, such as *agent*, or *runner*, impose more specific conditions than *subj*. Fourth, a dependency is associated in the semantic space with the operation of assigning an argument to a role of a lexical function. One word of a dependency will be taken as the lexical function and the other as its argument. The assignment is allowed if and only if the argument satisfies some semantic conditions (i.e., selection restrictions) required by the function.
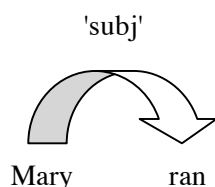
'subj'

Mary   ran

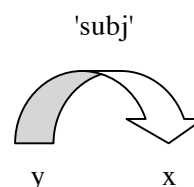**Figure 1: Subject dependency between "ran" (the head) and "Mary" (the dependent).**

'subj'

y   x
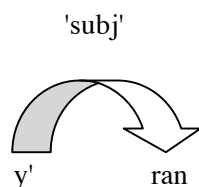
Figure 2: Subject dependency without specific words.

'subj'

y'   ran

Figure 3: Subject dependency with "ran" in the head position.
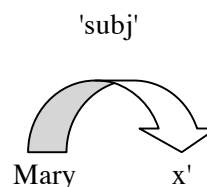
'subj'

Mary   x'

Figure 4: Subject dependency with "Mary" in the dependent position.

This paper will be focused on the way in which binary dependencies organise the operation of satisfying selection restrictions. We argue that, within a dependency, not only the specific head imposes restrictions on the words that can be assigned to the dependent role. But the reverse is also true. The specific dependent word imposes restrictions on the words that can be assigned to the head role. Figure 3 represents the *subj* dependency with a specific word, "ran", in the head position. Note that parameter *y'* is more specific than *y* in Figure 2. It stands for those nouns that can be the subject of verb "ran". The properties these nouns require to appear in such a position are, in fact, the selection restrictions imposed by the verb. Take now Figure 4. Here, the specific word is the noun "Mary", whereas *x'* stands for those verbs that have "Mary" as subject. Parameter *x'* is, then, more elaborated than *x*. It represents in fact the selection restrictions imposed by "Mary" to those verbs that can be assigned to the head role in the subject dependency.

Taking into account this description, our work is based on the assumption that a dependency carries at least two complementary selective functions, each one imposing its own selection restrictions. The main contribution of the paper is to describe how these restrictions can be used to simulate the process of linguistic disambiguation. We call *Optional Co-Composition* the disambiguation strategy relying on such restrictions. Dependencies will be put at the centre of the disambiguation process.

This article is organised as follows. Section 2 describes in model-theoretic terms the denotation of three linguistic categories: lexical words, dependencies, and lexico-syntactic patterns (as those illustrated in Figures 3 and 4). Special attention will be paid to the co-compositional properties underlying the lexico-syntactic patterns of dependencies. The next two sections will be devoted to explain how these properties are used to disambiguate word senses (Section 3) and structural ambiguity (Section 4). In fact, sections 3 and 4 are devoted to two case studies: the former is simply a proposal to word sense disambiguation and

incremental interpretation, while the latter is a corpus-based approach to attachment resolution. Even if they are very different in approach and method, these two case studies will be useful to understand how optional co-composition works.

2 Denotations

This section proposes a very broad-grained characterisation of the denotational space. Despite of being very generic, the level of description we use to define denotations will be enough to achieve the paper's aims. Our proposal will be focused on three basic linguistic categories:

– Lexical words such as "ran", "fast", "Mary", "program", "nice", etc.

– Syntactic dependencies such as "subj" (subject), "dobj" (direct object), "iobj_prep-name" (prepositional relation between a verb and a noun), "prep-name" (prepositional relation between two nouns), "attr" (attribute relation between a noun and an adjective), and so on.

– Lexico-syntactic patterns, which consist in combining a lexical word with a morpho-syntactic category by means of a dependency: "Mary + subj + VERB", "ADJ + attr + program", "nice + attr + NOUN", and so on.

Lexical words denote sets of properties while dependencies and lexico-syntactic patterns are defined as operations on these sets. We will pay more attention to the description of the operations.

2.1 Lexical Words

Since our objective is not to provide an exhaustive description of lexical words, we will only sketch some basic ideas. The different lexical categories, i.e., nouns, verbs, adverbs, and adjectives, are described in the same way: all of them are seen as sets of properties. This coarse-grained definition will be enough to shed light on our main goal, i.e., to describe the way in which lexical words are combined by binary dependencies. Concerning this goal, what we need to know is how two words are considered to be semantically compatible. Here, we follow the standard approach to selection restrictions: two words are compatible and, thereby, can be combined by a dependency if only if the sets they denote share some properties. For instance, the denotations of "Mary" and "ran" can be correctly combined by means of the subject dependency, because they have in common the properties that characterise the *runner* role, i.e. both denotations share a schematic individual with the ability to move fast. However, unlike most approaches to selection restrictions, we do not consider the verb as a compositional operator selecting for a compatible noun. Both verbs and nouns are sets of properties. The compositional operators that combine such properties will be associated to syntactic dependencies.

It is obvious that the semantic differences between two words belonging to different syntactic categories (e.g. "Mary" and "ran") cannot be entirely explained by enumerating the properties they do and do not share. These words are also different because they organise the properties they denote in very different ways. Yet, a study on denotations according to their various modes of organisation is beyond the scope of the paper. See a discussion on this issue in Gamallo (2003).

2.2 Dependencies

A dependency is a binary function which takes as arguments two sets of properties and gives as result a more restricted set. The two sets correspond to the denotations of the two words related by the dependency. We consider that the new set is the intersection of the two input sets, provided that there are no specific lexical constraints (meaning postulates) preventing the

intersection. The result of a dependency function is then by default an intersective interpretation. Let's take dependency *subj*. It is associated to two binary λ-expressions:

(1) $\lambda x \lambda y \ subj \ (x,y)$

(2) $\lambda y \lambda x \ subj \ (x,y)$

where *x* and *y* are variables for lexical denotations: *x* stands for the denotation of the head word while *y* represents the denotation of the dependent one. As in λ -calculus the left-to-right order of the binders is significant, function (1) is required when the denotation of the head word, *x*, is the first argument the function applies to. Likewise, function (2) must be used when it is first applied to the denotation of the dependent word, *y*. For the purpose of our work, the order of application is relevant. This will be discussed later. The application of either (1) or (2) to two specific word denotations results in a particular set intersection of the two arguments. Let's suppose that *mary* and *run* are the denotations of "Mary" and "ran", respectively (the verbal tense is not considered here). If one of the two binary functions, either (1) or (2), is applied to these two entities, we obtain:

(3) $subj \ (run,mary)$

which is a more restricted set of properties. Following Dependency Grammar, we assume that this set of properties must be seen as a new and more elaborate denotation of the head word, "ran". Consequently, (3) is the denotation of the verb "ran" in the head position of a subject dependency which assigns noun "Mary" to the dependent role. To abbreviate, we also may say that (3) is the denotation associated with "Mary ran" and other similar expressions.

As has been said before, we are not interested in enumerating the properties denoted by lexical words. Rather, our work will be focused on the combinatorial operations underlying every dependency. In particular, concerning an expression like "Mary ran", we are interested in the two different ways of reaching the composite denotation associated with that expression:

(4) $[\lambda x \lambda y \ subj \ (x,y)] \ (run)$
         $[\boldsymbol{\lambda y \ subj \ (run,y)}] \ (mary)$
         $subj \ (run,mary)$

(5) $[\lambda y \lambda x \ subj \ (x,y)] \ (mary)$
         $[\boldsymbol{\lambda x \ subj \ (x,mary)}] \ (run)$
         $subj \ (run,mary)$

Equation (4) relies on function (1). It represents a sequence of lambda applications which starts with the assignment of the verb denotation, *run*, to the head role of the *subj* dependency. The result of the first application is a unary function (in bold) than can be applied to word denotations like *mary*. This unary function can be seen as the linguistic constraints (syntactic subcategorisation + selection restrictions) the head *run* imposes on the word denotations appearing in the dependent position of the subject dependency. On the other hand, equation (5) is based on function (2). It starts by applying the binary dependency on the noun denotation, *mary*. It results in a unary function (in bold), which represents the linguistic constraints that *mary* imposes on the verb denotations appearing in the head position of that dependency.

2.3   Lexico-Syntactic Patterns

Even if the λ-expressions cited above does not explicitly contain any semantic element representing the syntactic categories of their arguments (i.e., the subcategorisation), we assume that such an information should be there. So, the *subj* dependency associated to (1)

and (2) is the object denoted by a syntactic pattern where the head and dependent words are syntactically constrained to be respectively a verb and a noun. According to this idea, a syntactic dependency like *subj* can be the function denoted by the following syntactic pattern: "NOUN + subj + VERB". Intuitively, when one of the two syntactic categories linked by the dependency is elaborated by a lexical unit, then we obtain a more specific pattern. This is what we call a "lexico-syntactic pattern". In the linguistic litterature (for instance, in Construction Grammar), there is a great variety of syntactic and lexico-syntactic patterns. Here, we are only interesting in those that are built by means of a binary dependency.

The unary function defined in (4) represents the denotation of the lexico-syntactic pattern "NOUN + subj + ran", whereas the unary function in (5) is the entity denoted by "Mary + subj + VERB". Notice that, as has been said before, the main syntactic categories (NOUN, VERB, or ADJ) are not distinguished in the semantic representation. They only appear in the syntactic space. Their inclusion in the semantic representation does not help us explain the basic principles underlying optional co-composition. The two unary functions denoted by the lexico-syntactic patterns take as argument a set of properties (a noun or a verb denotation) and return a more restricted set of properties. In both cases, the result is the same: *subj(run,mary)*. Yet, the choice of a compositional strategy is relevant. In (4), it's the verb denotation, *run*, that drives the combinatorial process since it is seen as the active functor imposing semantic conditions. In this functional application, *mary* is taken as a passive object filling those conditions. By contrast, in (5), *mary* is the active denotation that imposes specific conditions on the verbs appearing in the head position of the subject dependency. Consequently, the verb *run* is perceived here as a passive entity filling such conditions. As we will see in the next section, the choice between one of the two lexico-syntactic patterns relies on the degree of ambiguity associated to the two related words. In particular, we will assume that, within a syntactic dependency, the least ambigous word is used to discriminate the sense of the other one. For instance, let's suppose that the noun denotation *mary* is less polysemous than *run*. If it is true, the noun should be used to select a specific sense of the polysemic verb in the "subj" dependency. As *mary* designates someone with the ability to move, this noun denotation is able to select, among the different senses of *run*, the first sense appearing in WordNet, i.e., *someone moving fast*, while the seventh sense, *an engine working*, is prevented. So, in order for *mary* to be taken as the active discriminator, it would be suitable to choose the compositional strategy depicted in Equation (5). By choosing *mary* as sense selector, the verb denotation *run* is seen as the passive entity to be disambiguated. Likewise, if we combine *computer* with *run* using the same syntactic dependency (*subj*), the function denoted by the lexico-syntactic pattern "computer + subj + VERB" will enable to select the other verb sense, i.e., *an engine working*. We call this flexible procedure "optional co-composition".

2.4   Semantic Types

In model-theoretical terms, a lexical word denotation, for instance *computer* or *run*, is assigned a basic type. Let's note d such a basic lexical type. When the interpretation process require *computer* to be an active function selecting for a particular verb denotation in the subject position, then it will be combined with the dependency *subj*, whose type is $<<d,d>,d>$. As a result, we obtain the denotation of the lexico-syntactic pattern "computer + subj + VERB", which is a selective function of type $<d,d>$. A selective function activates a sense (or a set of senses) of a word by restricting its basic denotation.

Notice that such a compositional procedure cannot be considered as a particular instance of "type shifting" (cf. Partee 1987). Type shifting principles in Partee's conception only applies when the grammar fails to compose denotation types, i.e. when there is a type *mismatch*. According to this point of view, type shifting principles are seen as "last resort" mechanisms, which apply only when other compositional procedures fail. Optional co-composition, on the

contrary, is a general principle which must apply in every compositional interpretation. Moreover, unlike the type-shifting model, we are not required to change the basic semantic type of a lexical word when it behaves as a selective function. A word can change and denote a function, not as the result of type shifting, but after a combinatorial process. More precisely, the noun "computer" can be used as a selector functor of a particular sense of "ran", provided that the denotation of the noun combines with the subject dependency. So, word denotations evolve and *change* when they combine with dependencies in a higher semantic level of the compositional process. Thanks to optional co-composition, type-shifting is not required.

In the next two sections, we will describe the use of optional co-composition in two different disambiguation strategies: word sense disambiguation (Section 3) and corpus-based resolution of syntactic attachments (Section 4).

3 Optional Co-Composition and Word Sense Disambiguation

In most approaches to formal semantics, the interpretation process activated by composite expressions such as "Mary ran" or "nice program" relies on a rigid function-argument structure. The entity denoted by "Mary" is seen as the argument of the unary function denoted by "run", while the denotation of the adjective "nice" is a function that takes as argument the denotation of "program". Any syntactic dependency between two lexical words is generally represented in the semantic space as the assignment of an argument to a lexical function. To select the appropriate type of argument, the lexical function is provided with some linguistic constraints (i.e., selection restrictions). These restrictions not only allow the argument to be assigned to a role of the function role, but also they activate a specific sense of the argument. However, we claim that this rigid procedure is not the most appropriate way of dealing with word sense disambiguation (WSD). In the following subsection, we address the problems that arise when WSD is tackled using a rigid function-argument structure in the process of interpretation. To overcome such problems, Subsection 3.2 introduces the idea of co-composition. First, we analyse some drawbacks underlying the particular notion of co-composition proposed by Pustejovsky (3.2.1), and then, we propose a more general operation: "optional co-composition" (3.2.2). Finally, Subsection 3.3 will address the relation between optional co-composition and incremental interpretation.

3.1 Rigid Function-Argument Structure

According to many works on lexical semantics, the semantic representation of two words linked by a syntactic dependency is based on a function-argument structure. Each word is supposed to play a rigid and fixed role in that structure. One of the words is semantically represented as a selective function imposing constraints on the denotations of the words it combines with. The other word is seen as an argument filling the constraints imposed by the function. Besides the function-argument structure, a dependency also has a "head-dependent" organisation. Typically, when the "dependent" word is the argument, it is seen as the *complement* or *actant* of the "head" (see the analysis shown in Figure (5).
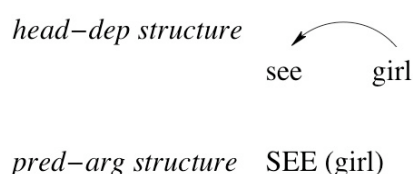
head−dep structure   see   girl

pred−arg structure   SEE (girl)

Figure 5: Complement organisation: "see (the) girl"

By contrast, if the "dependent" plays a more active role behaving more like a selective function, it is viewed as a *modifier* or *adjunct* (see Figure 6). In other words, the "dependent"

of the syntactic structure will be described either as a passive complement (an argument) if it satisfies the linguistic requirements of the head, or as an active modifier (a function) when it requires the head to be appropriate to its constraints.
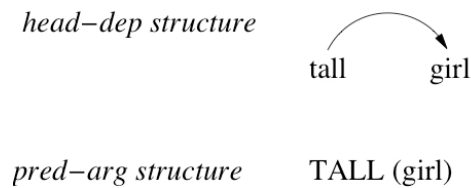
head−dep structure   tall → girl

pred−arg structure   TALL (girl)

Figure 6: Modifier organisation: "tall girl"

Most model-theoretical approaches propose semantic representations of word denotations that are in accordance with these ideas. According to these approaches, verbs and adjectives denote predicative functions, while nominals often refer to individuals assigned to those functions. However, such a rigid function-argument structure does not allow the nominal argument to impose lexical requirements on the verb or adjective. Consequently, it makes WSD less efficient. Take for instance the expression:

(6)   *long dinner*

According to the definitions of WordNet 2.0, both words are polysemous. On the one hand, "dinner" has two senses: it denotes a temporal event ("the dinner will be at seven"), and a mass object without dimensions ("light dinner"). On the other hand, adjective "long" has 10 senses. It denotes a primarily temporal dimension, a great height, a primarily spatial dimension, a phonetic property, and so on. In this section, we use the senses defined in WordNet because they are well-known and easily accessible to anyone and anywhere. We do not discuss here if WordNet is or is not a good model for sense representation. The purpose of this section is not to give a model of sense representation but to discuss how word senses are disambiguated, regardeless the way they are represented and organised.

Considering the fixed function-argument organisation associated with expression (6), the adjective must be taken as a function and the noun as its argument. So, selection restrictions are only imposed by "long" on the noun.

Let us propose a standard WSD algorithm based on a fixed function-argument structure. This basic algorithm was inspired by the discussion about the *sense enumeration approach* introduced by Pustejovsky (1995). The algorithm we propose consists of at least three subtasks: i) enumerate a list of lexical functions, ii) check selection restrictions, and iii) select the arguments satisfying those restrictions. Considering expression (6), the WSD procedure could be the following:

Step 1. Enumerating a list of lexical functions:

The 10 senses of the adjective "long", which are represented as 10 different lexical functions, are enumerated:

(7)   1. λx *Long* (x: <space>)
        2. λx *Long* (x: <time>)
        ...
        10. λx *Long* (x: <vowel>)

The properties *space*, *time*, etc. represent the selection restrictions imposed by the different functions on their argument.

Step 2. Checking restrictions:

Each function is applied to the ambiguity entity denoted by noun "dinner", which possess two different senses: *mass* and *time*. In each application, the predicate checks whether, at least, a sense of "dinner" satisfies the selection restrictions. This process is repeated until there are no more functions to check:

(8)   1.   [*λx Long* (*x: <space>*)]   (*dinner*: *<mass,time>*)

          2.   [*λx Long* (*x:* **<time>**)]   (*dinner*: *<mass,***time***>*)

          ...

          10.   [*λx Long* (*x: <vowel>*)]   (*dinner*: *<mass,time>*)

Step 3. Sense selection:

If the requirements are satisfied by a sense of the noun, the corresponding function is selected. In this example, only one functional application is allowed by the semantic requirements. It results in:

(9)   *Long* (*dinner: <time>*)

That is, we obtain at the end of the process a semantic representation involving the *time* sense of both "long" and "dinner".

This algorithm makes WSD very rigid: only the semantic requirements of a predetermined function allow one to select for the appropriate sense of the argument. Yet, the argument is not associated with requirements likely to be used to select for the appropriate adjective. Instead of selecting the correct sense of the adjective using semantic requirements imposed by the noun, this strategy needs to enumerate one by one the different senses of the adjective and check whether they can combine with the noun senses. It seems obvious that this enumeration procedure turns to be highly inefficient when the list of lexical functions corresponds to the most polysemic word. We argue that this WSD strategy is not suitable to simulate the understanding process. According to psychological evidences (cf. Justeson/Katz 1995), it seems to be more effective to always use the least ambiguous word as active disambiguator.

Let's analyse now the following expression:

(10)   *to drive the tunnel*

There is a more radical difference concerning the degree of polysemy conveyed by the two linked words. In WordNet, "drive" has 21 senses; one of them represents the mining situation. By contrast, "tunnel" merely has 2 related senses. In this case, it seems uncontroversial that the disambiguation procedure is activated by the least ambiguous word, i.e. the noun "tunnel", which selects for a specific verb sense: the action of mining. This process is both psychologically and computationally quicker than checking one by one the different senses of the verb. In order to simulate a more efficient disambiguation procedure, we need an operation allowing, when necessary, the nominal argument to be an active disambiguator.

3.2 Co-Composition

We assume that the two words related by a syntactic dependency impose semantic restrictions on each other. Not only verbs or adjectives must be taken as functions selecting different senses of noun denotations, but also nouns should be considered as active functors requiring different senses of verbs and adjectives. Before defining the idea of co-composition we propose in this article, let's analyse how the notion of co-composition was introduced in Generative Lexicon and then address the problems derived from such a notion.

3.2.1 Co-Composition in Generative Lexicon

Generative Lexicon (GL) claims that the lexical information conveyed by a nominal complement is able to shift the sense of the governing verb (cf. Pustejovsky 1995). Take the following expression:

(11)   *to drive the tunnel*

According to the GL's proposal, verb "paint" has a by default sense, namely *apply paint to a surface*, and a sense that is generated by composition: the making sense of the verb, i.e., *A painting is made by applying paint to a surface*. The making sense is not embedded in its lexical structure (or "qualia structure"), but in that of its potential nominal complements: e.g., "painting", "portrait", etc. Indeed, the qualia structure of these nouns specifies that the referred objects are artefacts made by the event of applying paint to a surface. When the denotation of the verb "paint" is applied to the denotation of a noun like "portrait", the two qualias are unified since they share part of their event information, in particular they have in common the event of *applying paint to a surface*. The event information resulting of this unification is that contained in the noun, namely *a portrait is made by applying paint to a surface*. Thanks to this unification, the verb takes a new sense, which was in fact previously stored in the qualia of its nominal complement. Table 1 illustrates this procedure. "[…]" symbolises other items of information contained in the qualia structure. To simplify the description, nothing is said about the specific ways of organising the lexical structure (e.g., qualia roles, event and argument structures, etc.). Notice that after combination only the qualia of the verb has been modified.

Word	Qualia structure	Qualia structure *after* combining "paint" with "portrait"
paint	[…] [*applying paint to a surface*] […]	[…] [*X is made by applying paint to a surface*] […]
portrait	[…] [*X is made by applying paint to a surface*] […]	[…] [*X is made by applying paint to a surface*] […]

Table 1: Partial definition of the qualia structure associated to "paint" and "portrait" (before and after combination)

Several problems arise from the co-composition operation such as it has been described in GL: it posits an unclear notion of "sense generation", it reduplicates information, it has a very limited coverage, and it inherits the usual rigidity from standard function application. Let's analyse these drawbacks.

First, GL does not define clearly the notion of "*generated sense*". According to the GL assumptions, the nominal complement is viewed as an active object endowed with the ability to generate a new sense of the governing verb. In expression (11), the complement makes the verb change the sense. As the verb did not have such a sense, GL assumes that a generative process took place. However, that sense is not completely new since it was within the noun qualia before starting the process of composition. It means that "sense generation" in GL is not the result of *creating* a new piece of information, but the result of *transferring* some existing information from the noun to the verb qualia. The difference between generation (such as it is described in GL) and selection seems to rely only on the place where some

specific lexical information is stored: when it is stored in the noun qualia, then the verb acquires a new sense after composition. The new sense is *generated* by transfer of information. By contrast, if it were stored in the verb qualia, then it would be necessary to define, not a generative operation, but an operation of *selecting* a verb sense using some information stored in the noun. In our approach, we assume a less controversial model of WSD. We presuppose that the combination of two words always activates a selective operation involving the process of satisfying selection restrictions. In addition, we consider that transfer information (or sense generation) is the natural consequence of intersecting (or unifying) the denotations of the dependent expressions. Unlike GL, we put the stress, not in information transfer, but in sense selection.

A second problem arises from polysemic expressions like the following:

(12)   *to paint a house*

This expression carries at least two senses: *apply paint to the surface of a house*, and *make a painting of a house*. To generate the making sense, GL is forced to provide the noun "house" with a very specific event information, namely, *(a painting of) X is made by applying paint to a surface*. This event is transferred to the qualia of "house" by inheritance from one of its potential hypernyms: the noun "painting". Note that this particular event information can compete with the making event, *X is made by the act of building*, which is inherited by "house" from its main hypernym: "artefact". According to GL, the verb "paint" takes the making sense from the qualia of "house" because this noun has been constrained by the verb to inherit the event *X is made by applying paint to a surface* from "painting". So, the noun "house" is able to *generate* the making sense of "paint" if and only if the verb has selected (or generated?) the painting sense of the noun before. This is not very intuitive. In fact, the event *X is made by applying paint to a surface*, which is very close to the by default sense of "paint", is transferred to the verb qualia by means of a noun, "house", whose qualia is very far from that event information. This event needs to be transferred twice: first, it is transferred by inheritance from "painting" to the qualia structure of "house". Then, it is transferred by qualia unification to the verb "paint". However, it would be more economic and intuitive if we decide to simply put this event directly within the qualia of "paint". Such a decision would prevent us from reduplicating the process of information transfer across thousands of noun qualias. In our approach, to interpret expression (12), we can follow a more economic approach: the noun "house", whose qualia contains a property like "being a material object", is provided with the suitable restrictions to select (and not generate) the making sense of the verb "paint". This selective operation only presupposes the following basic conditions: a house is a material object, it is possible to make a painting of any material object, and the process of making a painting is a piece of information contained in the qualia of the verb "paint". Once more, sense selection appears to be an appropriate model for word disambiguation.

There is a third problem. In GL, generative co-composition is triggered off if and only if the qualia of both the verb and the noun share some specific lexical information. The scope of this particular operation is then very narrow. It embraces very few cases: namely, the combination of some nouns with some verbs. GL does not make use of this operation to deal with polysemic "adjective + noun" combinations. These cases are analysed using an operation of selection, called "selective binding". By contrast, we make use of co-composition as a general semantic operation involved in the interpretation of every syntactic dependency between two words. In the next subsection, we will generalise the notion of co-composition so as to deal with all word dependencies.

Finally, there is a fourth problem. In GL, functional application relies on the standard approach to composition: verbs are taken as functions while nouns are taken as their

arguments. Co-composition is viewed as a unification operation that restricts the standard application of verbal functions to nominal arguments. Our model, however, does not follow such a rigid approach. Functional application will not be driven by relational words (verbs, adjectives, and adverbs), but by syntactic dependencies. In addition, we introduce the idea of optionality.

3.2.2   Optional Co-Composition

In this subsection, we will introduce a more general notion of co-composition than that presented above. We consider dependencies as active objects that control and regulate the selectional requirements imposed by the two related words. So, they are not taken here as merely passive syntactic cues related in a particular way (linking rules, syntactic-semantic mappings, syntactic assignments, etc.) to thematic roles or lexical entailments of verbs (cf. Dowty 1989). They are conceived of as the main functional operations taking part in the processes of sense interpretation and WSD.

On this basis, we associate functional application, not to relational expressions (verbs, adjectives, …), but to dependencies. A dependency, *dep*, can be defined as two binary λ-expressions:

(13) $\lambda x \lambda y \, dep \, (x,y)$

(14) $\lambda y \lambda x \, dep \, (x,y)$

As has been said in Section 2, these two functions represent two different options regarding the order of application:

> **Option 1**: Function (13) is first applied to the head, x, and then the resulting function is applied to y.

> **Option 2**: Function (14) is first applied to the dependent expression, y, and then the result is applied to x.

The process of interpreting a "head-dependent" expression by choosing only one of these two compositional options is what we call "optional co-composition". Such a mechanism makes WSD more flexible. Take again the compound "long dinner". The WSD algorithm we propose consists of the following steps:

Step 1. Identifying a dependency function:

As "long dinner" is an adjective+noun expression, the process starts by proposing the two optional functions associated with the attributive dependency:

(15) $\lambda x \lambda y \, attr \, (x,y)$

(16) $\lambda y \lambda x \, attr \, (x,y)$

Step 2. Choice of the least ambiguous word:

The dependency function is applied first to the word considered to be the best discriminator. By default, it will be the word with the least number of senses, that is, the least polysemic word. According to WordNet, the noun "dinner" is less ambiguous than "long". As the noun is the head of the composite expression, Function (15) is the compositional option that must be chosen. It is first applied to the denotation of "dinner":

(17) $[\lambda x \lambda y \, attr \, (x,y)]$ (*dinner*)
         $\lambda y \, attr \, (dinner, y)$

This is still a selective function likely to be applied to the word in the dependent position. Consequently, the denotation of the word "dinner" is taken here as the active functor. More precisely, this unary function can be seen as the denotation of the lexico-syntactic pattern "ADJ + attr + dinner". Basically, the remainding steps are the same as in the previous strategy.

Step 3. Enumerating the senses of the word discriminator:

The 2 senses of noun "dinner", which are represented as 2 different selective functions, are enumerated.

(18)   1.   *λy attr* (*dinner: <mass>, y: <mass>*)
       2.   *λy attr* (*dinner: <time>, y: <time>*)

Properties *time* and *mass* are seen as selection restrictions when they specify parameter *y*.

Step 4. Checking restrictions:

Each function is applied to the denotation of "long". Here, only two checking operations are activated.

(19)   1.   [*λy attr* (*dinner: <mass>, y: <mass>*)]   (*long: <space, time…, vowel>*)
       2.   [*λy attr* (*dinner: <time>, y: <time>*)]   (*long: <space, **time**…, vowel>*)

Step 5. Selecting senses:

The restrictions imposed by the temporal sense of "dinner" select the temporal sense of the adjective. The result is:

(20)   *attr* (*dinner*: *<time>, long*: *<time>*)

(20) represents the very specific denotation associated with "long dinner". This restricted denotation only contains the temporal properties of both *dinner* and *long*.

This strategy is more efficient than that defined in the previous section, since here the disambiguation process is controlled by the word that is considered to be the most appropriate to discriminate the sense of the other one. Optional co-composition makes functional application more flexible, since it allows to choose as selective function whatever word within a dependency, or even, if necessary, both words. Any word of a binary dependency may become an active functor and, then, be used to disambiguate the meaning of the other word.

Our semantic strategy is in accordance with the syntactic flexibility proposed by Cognitive Grammar (cf. Langacker 1991). This Theory considers that the meaning of a composite expression like "president of Ireland" can be built in two different ways. It can be the result of merging either the meaning of "president" with that of "of Ireland", or the meaning of "president of" with that of "Ireland". Therefore, Langacker also postulates two potential ways of combining the meaning of two dependent expressions. However, unlike our proposal, this flexibility is not used to deal with WSD. He merely uses it to explain some semantic issues at the discourse level. So, our approach may serve to expand the Langacker's framework.

3.3   Further Discussion: Incremental WSD

WSD can be integrated into a more general process: incremental interpretation. The aim of this subsection is to describe how optional co-composition disambiguates the sense of words within an incremental framework.

3.4   The Incrementality Hypothesis

One of the basic assumptions on interpretation made in frameworks such as Dynamic Logic (cf. Groenedijk/Stokhof 1991), Discourse Representation Theory (cf. Kamp/Reyle 2003), and Situation Semantics (cf. Barwise/Perry 1987), is that the meaning of a sentence both is

dependent of the meaning of the previous sentence in the discourse, and modifies itself the meaning of the following sentence. Sentence meaning does not exist out of discursive unfolding. Meaning is incrementally constructed at the same time as discourse information is processed.

We assume that incrementality is true not only at the inter-sentence level but also at the inter-word level, i.e., between dependent words. In order for a sentence-level interpretation to be attained, dependencies must be established between individual constituents as soon as possible. This claim is assumed by a great variety of research (Kempson et al. 2001, Kempson et al. 1997, Milward 1992, Costa et al. 2001, Schlesewsky/Bornkessel 2004). The incrementality hypothesis states that information is built up on a left-to-right word-by-word basis in the interpretation process (cf. research (Kempson et al. 2001). The meaning of an utterance is progressively built up as the words come in. The sense of a word is provided as part of the context for processing each subsequent word. Incremental processing assumes that humans interpret language without reaching the end of the input sentence, that is they are able to assign a sense to the initial left fragment of an utterance. This hypothesis has received a large experimental support in the psycholinguistic community over the years (McRae et al. 1997, Tanenhaus/Carlson 1989, Truswell et al. 1994).

A left-to-right interpretation process can be viewed as the semantic counterpart of the syntactic analysis proposed by Word Grammar (cf. Hudson 2003), and adapted by other dependency grammars. According to this theory, a syntactic structure puts the words in left-right order and links them by arcs that carry labels. The graphic representation of such a linear structure for the expression:

(21)   *to finish the long construction*

is given by Figure 7. Following the incrementality assumption, we argue that the order of application of dependency functions on word denotations is from left to right. To interpret (21), dependency *dobj* is interpreted before *attr*. The meaning of words is gradually elaborated as they are syntactically integrated in new dependencies. Syntactic analysis and semantic interpretation are merged into the same incremental process of information growth.
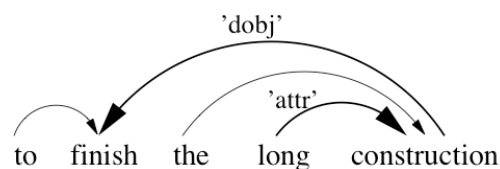

Figure 7: WG analysis of "to finish the long construction"

3.5   Incremental WSD and Optional Co-Composition

WSD evolves dependency-by-dependency from left-to-right. In this evolutionary process, the word taken as sense discriminator within a dependency can be the one whose denotation has been semantically specified in the previous dependencies. Take again expression (21). In WordNet, the noun "construction" carries 7 senses. It denotes among them: *the temporal act of constructing* and *the material thing constructed*. As "long" can refer, among its 10 senses, to both temporal and material objects, the attributive expression "long construction" remains ambiguous; it can mean either a long action or a long physical object. However, within the larger verbal expression (21), incremental disambiguation leads to only one sense, namely the temporal one: *the act of construction is long*. Let us describe quickly this process:

First Dependency: "finish" disambiguates "construction".
In the *dobj* dependency, the verb "finish" requires a temporal complement. So, in this particular context, noun "construction" denotes the act of constructing.

Second Dependency: "construction" disambiguates "long".
The disambiguated noun, "construction", is used as the word discriminator within the *attr* dependency. Consequently, it is taken as the active function selecting for the temporal sense of adjective "long".

Following the optional co-composition strategy, in the second dependency the word discriminator is not the very ambiguous adjective "long". This word is disambiguated by the denotation of the noun "construction", which has been elaborated and disambiguated in the first dependency. So, in the second dependency the discriminator is the temporal sense of the noun. Co-compositionality makes easy use of contextual information available from the previous dependencies to immediately solve the problem of lexical ambiguity. Such an incremental process cannot be easily simulated within a rigid approach to the function-argument structure.

4   Optional Co-Composition and Syntactic Attachment Resolution

The previous section describes an approach to WSD based on optional co-composition. However, as the notion of word sense remains beyond our work, we were not able to afford any computational and realistic experiment using the algorithm defined before in 3.2.2. By contrast, in this section, we briefly describe a computational strategy to solve another type of ambiguity: structural ambiguity. This strategy does not rely on controversial tasks such as enumerating the sense of words in an accurate way. It consists in solving syntactic attachment using selection restrictions learnt from large corpora. This realistic strategy consists of three steps. First, we automatically acquire selection restrictions from corpora (Subsection 4.1). Then, the acquired information is used to build a lexicon with information on lexico-syntactic restrictions (Subsection 4.2). Taking into account this corpus-based lexicon, a syntactic attachment heuristic is proposed (Subsection 4.3). The attachment heuristic relies on the assumption on optional co-composition. Finally, some results are evaluated (4.4). In this section, we only provide some basic ideas on the extraction method. Our aim is to understand the process at stake in this case study, namely attachment resolution. The extraction method has been accurately described in (Gamallo et al. 2005).

4.1   Selection Restrictions Acquisition

An experiment to automatically acquire selection restrictions was performed on a Portuguese corpus. We used an unsupervised and knowledge-poor method. It is unsupervised because no training corpora semantically labelled and corrected by hand is needed. It is knowledge-poor since no handcrafted thesaurus such as WordNet and no Machine Readable Dictionary is required (cf. Grefenstette 1994). The method consists of the following sequential processes. First, raw Portuguese text is automatically tagged and then analysed in binary syntactic dependencies using a simple heuristic based on Right Association. For instance, the expression *salário do secretário* ("salary of the secretary") is analysed as a binary dependency:

(22)   *of* (*salary*,*secretary*)

Second, following the basic idea behind co-composition, we extract two lexico-syntactic patterns from every binary dependency. From (22), we extract both "salary of NOUN" and "NOUN of secretary", which are represented as selective functions:

(23) $\lambda y \, of \, (salary, y)$

(24) $\lambda x \, of \, (x, secretary)$

Third, we generate clusters of similar lexico-syntactic patterns by computing their word distribution. We assume, in particular, that two lexico-syntactic patterns are considered to be similar if they impose the same selection restrictions. It is also assumed that two patterns impose the same selection restrictions if they co-occur with the same words. Similarity between lexico-syntactic patterns is calculated by using a particular version of the Lin coefficient (cf. Lin 1998). A cluster of similar patterns is given in (25) below:

(25) $\lambda y \, of \, (salary, y)$
        $\lambda y \, of \, (post, y)$
        $\lambda y \, subj \, (resign, y)$
        $\lambda x \, of \, (x, competent)$

which is a cluster containing the pattern (23). Finally, each cluster is associated with those words co-occurring at least once with most ($\geq 80\%$) of the patterns constituting the cluster. For instance, cluster (25) is associated with the following list of nouns:

secretary, president, minister, manager, worker, journalist

This word list is used to extensionally define the selection restrictions imposed by the patterns aggregated in the cluster (25). In fact, the list of words required by similar patterns represents the extensional description of their semantic preferences. Such a cluster is, thus, a particular way of representing a word sense.

4.2   Building a Lexicon with Lexico-Syntactic Restrictions

secretário ('secretary')

– $\lambda x \, de \, (x, secretário)$ / $\lambda x \, of \, (x, secretary)$ =
cargo, carreira, categoria, competência, escalão, estatuto, função, remuneração, trabalho, vencimento
('post', 'career', 'category', 'qualification', 'rank', 'status', 'function', 'remuneration', 'job', 'salary')

– $\lambda y \, de \, (secretário, y)$ / $\lambda y \, of \, (secretary, y)$ =
administração, assembleia, autoridade, conselho, direcção, empresa, entidade, estado, governo, instituto, juiz, ministro, ministério, presidente, serviço, tribunal órgão
('administration', 'assembly', 'authority', 'council direction', 'company', 'entity', 'state', 'government', 'institute', 'judge', 'minister', 'ministery', 'president', 'service', 'tribunal organ')

– $\lambda x \, iobj\_a \, (x, secretário)$ / $\lambda x \, iobj\_to \, (x, secretary)$ =
aludir, aplicar:refl, atender, atribuir, concernir, corresponder, determinar, presidir, recorrer, referir:refl, respeitar
('allude', 'apply', 'attend', 'assign', 'concern', 'correspond', 'determine', 'resort', 'refer', 'relate')

– $\lambda x \, iobj\_a \, (x, secretário)$ / $\lambda x \, iobj\_to \, (x, secretary)$ =
caber, competir, conceder:vpp, conferir, confiar:vpp, dirigir, incumbir, pertencer
('concern', 'be-incombent', 'concede', 'confer', 'trust', 'send', 'be-incombent', 'belong')

– $\lambda x \, iobj\_por \, (x, secretário)$ / $\lambda x \, iobj\_by \, (x, secretary)$ =
assinar, conceder, conferir, homologar, louvar, subscrever
('sign', 'concede', 'confer', 'homologate', 'compliment', 'subscribe')

– $\lambda x \, subj \, (x, secretário)$ / $\lambda x \, subj \, (x, secretary)$ =
definir, estabelecer, fazer, fixar, indicar, prever, referir
('define', 'establish', 'make', 'fix', 'indicate', 'foresee', 'refer')

Table 2: Excerpt of lexicon entry *secretário*

The acquired clusters of patterns and their associated words are used to build a lexicon with syntactic and semantic subcategorisation information. Table 2 shows an excerpt of the

information the system learnt about the noun *secretário* ("secretary"). This noun is associated with six different lexico-syntactic patterns. Notice that most of the patterns in Table 2 does not denote standard lexical functions. For instance, the unary function *λx subj*(*x,secretário*) selects for those verbs – *definir* ("define"), *estabelecer* ("establish"), etc. – that have the noun *secretário* as subject. Such non-standard lexical functions are in accordance with the notion of co-composition. This is a significant novelty of our approach.

4.3   Attachment Heuristic

Optional co-composition is at the centre of syntactic attachment. Indeed, the heuristic we use to check if two words are or are not syntactically attached is mainly based on the notion of optional co-composition. This heuristic states that two words, *w*1 and *w*2, are syntactically attached by a dependency, *dep*, if and only if the following algorithm is verified:

Option 1   Check the Head Requirements:
                if
                λy dep(*w*1,*y*) requires *w*2
                then
                λy dep(*w*1,*w*2) is a true attachment
                else
Option 2   Check the Dependent Requirements
                if
                λx dep(*x,w*2) requires *w*1
                then
                λy dep(*w*1,*w*2) is a true attachment

In order to check the head and dependent requirements, we use the information stored in the lexicon. Let's see an example. Take the expression:

(26)   *compete ao secretário do ministro* ("is incumbent on the secretary of the minister")

There exist at least three attachment candidates: 1) *competir* ("be incumbent") is attached to *secretário* ("secretary") by means of preposition *a* ("to"); 2) *competir* is attached to *ministro* ("minister") by means of preposition *de* ("of"); 3) *secretário* is attached to *ministro* by means of preposition de. Yet, only two of them are correct. To select the correct attachments, we make use of both the selection restrictions stored in the lexicon and the heuristic described above. Let's suppose our corpus-based lexicon is only constituted by the entry secretário (Table 2). Such a lexicon does not contain any information on *competir* and *ministro*. The system verifies whether the first attachment candidate is true. According to the lexico-syntactic information illustrated in Table 2, the attachment is allowed because the dependent requirements are satisfied:

            λx iobj_a (*x,secretário*) requires *competir*

Then, the system verifies the second possible attachment. The attachment is correct since the head requirements are satisfied:

            λy de (*secretário,y*) requires *ministro*

Finally, the third candidate is checked. In this case, there is no attachment because the lexicon has not any subcategorisation information on the two considered words: *competir* and *ministro*. So, the final syntactic analysis of expression (26) is represented by the two following dependencies:

            iobj_a (*competir,secretário*)
            de (*secretário,ministro*)

The graphic representation of this analysis is given by Figure (8). Consequently, solving syntactic attachments is somehow a semantic task mainly driven by dependency functions.
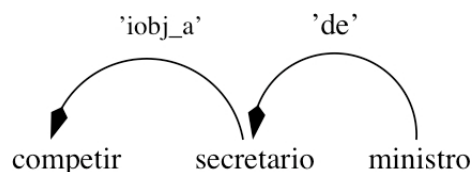
Figure 8: DG analysis of *compete ao secretário do ministro*
("is incumbent on the secretary of the minister")

Notice that, unlike most approaches to attachment resolution, the restrictions imposed by verbs are not necessary conditions to check whether the nominal complements are attached to the head. Thanks to co-compositional information, we can make use of the requirements imposed by the nominal complements on verbs. This is particularly useful if the lexicon (and the corpus from which the lexicon was extracted) is not endowed with verb restrictions. So, if the lexicon does not provide specific subcategorisation information on a particular verb, we can use noun requirements to check whether that verb is attached or not to its candidate complements.

In sum, the attachment heuristic based on optional co-composition allows us to enhance the parsing task, in particular, the resolution of syntactic ambiguity. Experimental results on the evaluation procedure will be reported in the following subsection.

4.4   Evaluation

Table 3 below reports the test scores concerning the precision and recall of the experiments performed on 1266 test expressions, which were randomly selected from a test corpus. Each test expression was constituted by a sequence of three basic phrases (or chunks). The types of sequences considered for evaluation were the following: (*vp,np,pp*), (*vp,pp,pp*), and (*np,pp,pp*). They are ambiguous sequences since three attachment candidates can be proposed but only two of them are correct. Given a sequence of phrases, we made use of our attachment heuristic to select the correct dependencies between the words heading each basic phrase.

BASELINE			
sequences	**Prec. (%)**	**Rec. (%)**	**F-S (%)**
np,pp,pp	71	71	71
vp,pp,pp	83	83	83
vp,np,pp	75	75	75
Total	76	76	76
OPTIONAL CO-COMPOSITION			
sequences	**Prec. (%)**	**Rec. (%)**	**F-S (%)**
np,pp,pp	85	73	79
vp,np,pp	92	75	83
vp,pp,pp	86	70	77
Total	88	73	80

Table 3: Evaluation of Attachment Resolution

We made a comparison between our method and a baseline strategy. As a baseline, we used the attachments proposed by Right Association. That is, for each expression of the test data, this strategy always proposes the attachment by proximity, that is: *phrase1* is attached to *phrase2*, *phrase2* is attached to *phrase3*, and *phrase1* is not attached to *phrase3*. The values

plotted in Table 3 shows that the results performed using the heuristic based on optional co-composition are significatively better concerning precision. As far as recall is concerned, our approach remains lower than the baseline since the subcategorisation lexicon is still incomplete. In order to improve recall, we need to allow the clustering strategy to generate more lexical entries. This will provide the lexicon with more information on lexico-syntactic requirements.

4.5 Further Discussion: Other Corpus-Based Applications

The fact that any binary dependency contains two complementary lexico-syntactic patterns and, then, two different types of semantic requirements is not new. In (Gamallo et al. 2001), we used such patterns as word contexts in order to automatically measure the semantic similarity between word pairs. The results were compared to those obtained by the corpus-based strategy defined by Grefenstette (1994), who only used as local contexts standard lexico-syntactic patterns. For instance, according to Grefenstette, the expression "president of Ireland" gives rise to only the pattern "NOUN of Ireland". From this viewpoint, pattern "president of NOUN" cannot be considered because "president of" is not a well-formed expression. Following the Constitutive Grammar framework, Grefenstette does not associate any syntactic category to this expression. The results reported in (Gamallo et al. 2001) showed that the application using co-compositional lexico-syntactic patterns performed better than that defined by Grefenstette (who followed Constitutive Grammar). Other corpus-based approaches to measure word similarity also had promising results using this type of patterns (cf. Lin 1998, Reinberger/Daelemans 2003). Such successful applications show that co-compositional patterns are semantically discriminant and, then, useful for finding further semantic regularities. It is true that these experiments are not an irrefutable argument proving that nouns select for verbs in the same way that verbs select for nouns. However, they can be considered as a first approximation to validate the main theoretical assumption we made in this paper.

5 Conclusions and Some Final Remarks

According to most semantic approaches, two syntactically related words are interpreted as a fixed "function-argument" pair, where each word has always a pre-defined role: the one is the selective function imposing semantic requirements, the other is the passive argument filling them. We addressed here a different viewpoint. Each related word can be perceived as a functor and an argument alternatively. The word that will be construed as the selective function in a particular dependency must be the one whose denotation is the most elaborate (the least ambiguous word). It will be used to disambiguate the word considered as being the most ambiguous, and which has been taken as argument.

This strategy attempts to simulate how a hearer/reader disambiguates words. We claim that it is a psychologically motivated strategy. Furthermore, it should also reduce computational complexity.

In this paper, syntactic dependencies were endowed with a combinatorial meaning. The fact of characterising dependencies as compositional devices has important consequences on the way in which the process of semantic interpretation is seen. Some of these consequences are the following.

First, semantic interpretation is conceived as a mechanism that processes in parallel several linguistic resources, namely syntactic and lexical information. This assumption lies on a psycholinguistic hypothesis reported in several (McRae et al. 1997, Tanenhaus/Carlson 1989, Truswell et al. 1994). According to this hypothesis, the analysis of an utterance depends on the simultaneous access to syntactic and lexical (and maybe pragmatic) information. The

parallel approach is opposed to those frameworks based on the syntactic autonomy (cf. Frazier 1989). Inspired by chomskian generativism, such approaches assume that the semantic interpretation can be activated only if the syntactic processing has been achieved in a previous stage. This way, lexical information only becomes accessible after having built the syntactic structure of the sentence. According to the parallel approach, however, language processing must immediately exploit the full range of potentially available lexical knowledge in order to reduce meaning and structural indeterminacy. The way we have defined dependencies is in accordance with this approach.

Another consequence of our viewpoint on dependencies is the treatment of verb alternations. Polysemous verbs have various subcategorisation frames, which correspond to their different senses. In our approach, all aspects of the subcategorisation frames of a verb - namely, number of complements, semantic roles involved, type of participants - are accessed and elaborated dependency by dependency during the interpretation process. Subcategorisation frames are taken here as pieces of lexical information used by dependencies as restrictions to select the specific sense of words. By contrast, in many linguistic approaches, each subcategorisation frame of a verb denotes a particular function. In these approaches, the process of selecting a specific subcategorisation frame of the verb (that is, disambiguating the verb) is a task that must be achieved before starting the process of interpretation. To select a specific verb sense, the syntactic analysis is constrained by some lexical and linking rules (cf. Davis 1996). These rules use some aspects of the lexical information available from the verb and its complements to map a particular verb frame onto the surface of syntactic structure. So, linking rules make it possible to select the correct verb sense (with the correct type of complements) before starting semantic interpretation. However, some not trivial questions arise: how could one distinguish the lexical information used by linking rules to constrain syntactic analysis from that used by the semantic module to reach a specific interpretation? What is the descriptive goal of a theory of meaning: either to describe the lexical constraints involved in syntactic analysis, or rather to study the semantic interpretation process? Such questions would not arise if we consider that syntactic analysis and meaning interpretation are one and the same semantic procedure. Our proposal on the meaning of dependencies characterises both syntactic analysis and semantic interpretation as the same language process.

Finally, the most direct consequence of our work is that dependencies are semantic operations driving the process of solving lexical and syntactic ambiguity. This is in accordance with one of the main assumptions of Cognitive Grammar (cf. Langacker 1991). Within this framework, syntactic and lexical information are not two different types of linguistic information. Morpho-syntactic categories, syntactic relations, word senses, and lexical properties are seen as semantic entities. Whereas the lexicon contains very specific semantic information, Syntax is seen as a set of semantic categories and properties defined at a very abstract and schematic level. So, according to this viewpoint on Semantics, the lexical and syntactic disambiguation strategies proposed in this paper should be conceived as two particular instances of a more general approach to semantic disambiguation. In sum, our work belongs to the linguistic tradition that puts Semantics at the core of linguistic analysis.

References

Barwise, Jon/Perry J. (1987): *Recent Developments in Situation Semantics. Language and Artificial Intelligence*. Berlin.

Costa, Fabrizio/Lombardo, Vincenzo/Frasconi, Paolo/Soda, Giovanni (2001): "Wide coverage incremental parsing by learning attachment preferences". In: Esposito, Floriana (ed.): *Advances in Artificial Intelligence. 7th Conference of the Italian Association for Artificial Intelligence (AI*IA 2001)*: 297–307.

Davis, Anthony (1996): *Lexical semantics and linking in the hierarchical lexicon*. PhD thesis, Standford University.

Dowty, David (1989): "On the semantic content of the notion of thematic role". *Properties, Types, and Meaning* 2: 69–130.

Frazier, Lyn (1989): "Against lexical generation of syntax". In: Marslen-Wilson, William D. (ed.): *Lexical Representation and Process*. Cambridge, MA: 505–528.

Gamallo, Pablo (2003): "Cognitive characterisation of basic grammatical structures". *Pragmatics and Cognition* 11/2: 209–240.

Gamallo, Pablo/Agustini, Alexandre/Lopes, Gabriel (2005): "Clustering syntactic positions with similar semantic requirements". *Computational Linguistics* 31/1: 107–146.

Gamallo, Pablo et al. (2001): "Syntactic-based methods for measuring word similarity". In: Mautner, Vàklav/Moucek, Roman/Moucek, Pavel (eds.): *Text, Speech, and Discourse (TSD-2001)*. Berlin: 116–125.

Grefenstette, Gregory (1994): *Explorations in Automatic Thesaurus Discovery*. Norwell, MA. (= *The Springer International Series in Engineering and Computer Science* 278).

Hudson, Richard (2003): "The psychological reality of syntactic dependency relations". In: Kahane, S./Nasr, A. (eds.): *MTT 2003. Proceedings of the First International Conference on Meaning-Text Theory*. Paris: 181–192.

Justeson, Jon/Katz, Slava (1995): "Principled disambiguation. Discriminating adjective senses with modified nouns". *Computational Linguistics* 21/1: 1–27.

Kahane, Sylvain (2003): "Meaning-text theory". In: Ágel, Vilmos et al. (eds.): *Dependency and Valency. An International Handbook of Contemporary Research*. Berlin.

Kamp, Hans/Reyle, Uwe (1993): *From Discourse to Logic. Introduction to Model-theoretic Semantics of Natural Langue. Formal Logic and Discourse Representation Theory*. Dordrecht.

Kempson, Ruth/Meyer-Viol, Wilfried/Gabbay, Dov (1997): "Language understanding. A procedural perspective". In: Retoré, Christian (ed.): *First International Conference on Logical Aspects of Computational Linguistics*. London: 228–247. (= *Lecture Notes in Artificial Intelligence* 1328).

Kempson, Ruth/Meyer-Viol, Wilfried/Gabbay, Dov (2001): *Dynamic Syntax. The Flow of Language Understanding*. Oxford.

Langacker, Ronald W. (1991): *Foundations of Cognitive Grammar. Descriptive Applications*. Vol. 2. Stanford.

Lin, Dekang (1998): "Automatic retrieval and clustering of similar words". In: *Proceedings of the 17th international conference on Computational Linguistics (COLING-ACL'98)*. Montreal/Quebec: 168–774.

McRae, Ken/Ferreti, Todd/Amoyte, Liane (1997): "Thematic roles as verb-specific concepts". In: MacDonald, Maryellen (ed.): *Lexical Representations and Sentence Processing*. Hove: 137–176. (= *Language and Cognitive Processes* 12/2–3).

Milward, David (1992): "Dynamics, dependency grammar and incremental interpretation". In: *Proceedings of the 15th Conference on Computational Linguistics (Coling'92)*. Vol. 4. Nantes: 1095–1099.

Partee, Barbara (1987): "Noun phrase interpretation and type-shifting principles". In: Groenendijk, Jeroen/de Jongh, Dick/Stokhof, Martin (eds.): *Studies in discourse representation theory and the theory of generalized quantifiers*. Dordrecht: 115–143.

Pustejovsky, James (1995): *The Generative Lexicon*. Cambridge.

Reinberger, Marie-Laure/Daelemans, Walter (2003): "Is shallow parsing useful for unsupervised learning of semantic clusters?" In: Gelbukh, Alexander F.: *Proceedings of the 4th Conference on Intelligent Text Processing and Computational Linguistics (CICLing-03)*. Mexico City: 304–312.

Schlesewsky, Mathias/Bornkessel, Ina (2004): "On incremental interpretation. Degrees of meaning accessed during sentence comprehension". *Lingua* 114: 1213–1234.

Tanenhaus, Michael K./Carlson, Grez N. (1989): "Lexical structure and language comprehension". In: Marslen-Wilson, William (ed.): *Lexical Representation and Process*. Cambridge, MA/London: 530–561.

Tesnière, Lucien (1959): *Eléments de syntaxe structurale*. Paris.

Truswell, Jonh C./Tanenhaus, Michael K./Garnsey, Susan M. (1994): "Semantic influences on parsing. Use of thematic role information in syntactic ambiguity resolution". *Journal of Memory and Language* 33: 285–318.