A step towards incremental generation of logical forms

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
Gabriel.Bes@univ-bpclermont.fr

Abstract

This paper presents AsdeCopas, a module designed to interface syntax and semantics. AsdeCopas is based on hierarchically organised semantic rules, that output formulas in a flat language. In this paper, we show how this system can be used in the following applications: a) semantic disambiguation; b) logical formulas construction (in Minimal Recursion Semantics); c) question interpretation.

1 Introduction

We present AsdeCopas, a syntax-semantic interface based on hierarchically organised rules.

AsdeCopas is integrated in a system where the input text is first transformed into a graph and then passed to AsdeCopas. AsdeCopas can be used in several ways.

It can be used to enrich the graph (Figure 1), for example, by labeling its arrows.

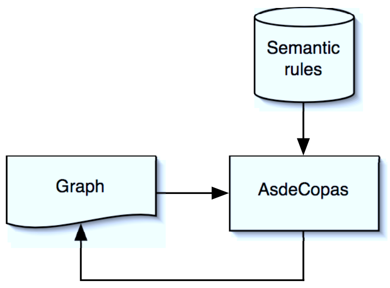

Figure 1: Enriching the graph

It can be used in a desambiguation process and to generate logical formulas. In this paper we show how AsdeCopas can be used to choose between several semantic values of some quantifiers and also how it can generate underspecified formulas in Minimal Recursion Semantics (MRS) (Copestake et al., 2001). Additionally, it can be used to add constraints to these underspecified formulas. As AsdeCopas makes a controled generation of variables, these new formulas can be simply added to the previous underspecified MRS formulas and the rules responsible for generating MRS underspecified structures remain unchangeable.

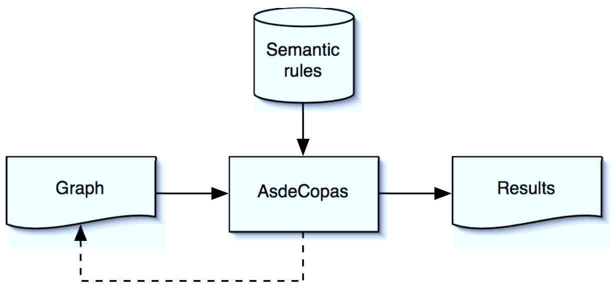

Figure 2: Desambiguation

Notice that in all the applications, AsdeCopas could previously be used to enrich the graph and the rules used in each task should take it into consideration.

This paper is organised as follows: we start with the motivation for this work. Then, in section 3, we present our approach. This includes a description of the semantic rules formalism and a brief overview of the algorithm behind AsdeCopas. In section 4 we introduce some applications. Final remarks and future directions can be found in section 5.

2 Motivation

In 1992, an exhaustive study of the Portuguese tourist resources was made by the Direcção Geral de Turismo (DGT) and afterwards the Inventory of Tourist Resources (IRT) emerged. In order to access it, multimedia "kiosks" were developed and a system called Edite (da Silva, 1997; Reis et al., 1997; da Silva et al., 1997) was created with the purpose of being integrated in these "kiosks" and to allow database access using natural language. Edite had a set of linguistically motivated traditional modules (semantic rules associated with syntactic rules, bottom-up parser, and so on) and it soon became saturated: adding a new syntactic rule caused dramatic side effects, a new semantic value could duplicate the number of generated formulas, etc. It was this experiment that made us change our approach and invest in a more robust methodology. We found in the 5P Paradigm (Bès, 1999; Bès and Hagège, 2001; Hagège, 2000) the background we were looking for and the syntax-semantic interface presented in this paper reflects the effort of adapting to a more robust methodology.

Many systems base their interface in rules, that according to (Aït-Mokhtar et al., 2002) "encode hypothetical local interpretations of sub-strings, yet to be validated by the production of a full parse". This is typically what happens to syntactic-semantic bottom-up parsers where each semantic rule is associated with a syntactic rule. Even if these systems do not fail when a sub-string interpretation fails, their parsers need to deal with a combinatory explosion of multiple interpretations of words, even if syntactic conditions would allow precise values to be chosen. This is due to the fact that at each step there is not a whole vision of the (syntactic) context. An additional effect of not having access to context is that spurious ambiguities are produced.

As an example, consider the Portuguese word *qualquer* (roughly, *any*), which can take several semantic values (see (Móia, 1992) for a detailed discussion about the multiple interpretations of *qualquer*):

- In *Qualquer cão gosta de ossos* (*All dogs like bones*) it has an universal value (`univ`);
- In *Ele tem qualquer problema* (*There is some problem with him*) it has an existential value (`exist`);
- In *Ele é um jornalista qualquer* (*He is an insignificant journalist*) it is an adjective, and it means something like *with no relevant characteristics in the class denoted by the noun it qualifies*. We will denote this semantic value as `indiscriminate`;
- In *Ele não é um jornalista qualquer* (*He is not an insignificant journalist*) it has the same `indiscriminate` value.

Let us assume that on the right of a main verb in the scope of negation, *qualquer* can only take the `indiscriminate` semantic value. Typically, in a bottom-up parsing (Figure 3) we will not be able to discard unnecessary values, as in point (1), when finally we have the whole vision of the subtree, the semantic rule will not take into consideration the negation inside V.

```
S -> NP VP
VP -> V NP (1)
V -> neg v | v | ...
neg -> não
v -> é | ...
NP -> art n qt | art n | ...
art -> um | ...
n -> jornalista | ...
qt -> qualquer | ...
```

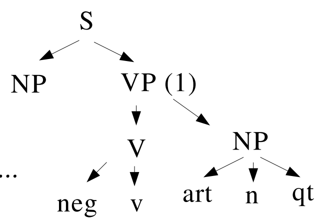

Figure 3: Grammar and *qualquer* example

Another kind of interface can be found in systems such as ExtrAns (Mollá et al., 2003), where the syntax-semantic interface is executed over dependencies. According to (Mollá and Hutchinson, 2002), the current version of ExtrAns uses either Link Grammar or the Conexor FDG parser.

In the first situation, the logical-form is constructed by a top-down procedure, starting in the head of the main dependency and following dependencies. The algorithm is prepared to deal with a certain type of dependencies, and whenever an unexpected link appears, a special recovery treatment is applied. When describing the algorithm, the authors say that most of these steps "... become very complex, sometimes involving recursive applications of the algorithm" and also that "specific particularities of the dependency structures returned by Link Grammar add complexity to this process" (Mollá and Hutchinson, 2002).

In the Conexor FDG case, the bottom up parser used has three stages. In the first one (introspection) possible underspecified predicates are associated with each word. Object predicates introduce their own arguments, but other predicates remain incomplete until the second stage (extrospection). During extrospection, arguments are filled by examing the relation between each word and its head. Sometimes dummy arguments need to be assigned when the algorithm faces disconnected dependency structures. A third stage (re-interpretation) is needed to re-analyse some logical constructs. According to the authors, the algorithm can-not produce the correct argument structure for long distance dependencies.

As we will see, within AsdeCopas:

- rules allow to identify semantic values that depend on the context;
- the algorithm itself is independent from the utilised dependency structures. Only semantic rules have to be adapted to the dependency structures;
- there is no need to recursively apply the algorithm or to create dummy arguments due to disconnected dependency structures: in these situations, default rules are triggered;
- long distance dependencies cause no problem as rules are sensitive for the (possibly non-local) syntactic context;
- all words, independently from their category, are mapped into formulas in one step: since rules are self-contained, they contain all the necessary information to calculate the corresponding formula.

3 Our approach

3.1 Brief overview

Aït-Mokhtar (Aït-Mokhtar et al., 2002) defines an incremental rule as "a self-contained operation, whose result depends on the set of contextual restrictions stated in the rule itself. [...] If a sub-string matches the contextual restrictions, the corresponding operation applies without later backtracking" . This is the gold property we achieved for our semantic rules.

Now the question is: how are we going to define an incremental rule if in our output we have predicates sharing variables, scope relations to define, and so on? We propose a solution based on the following:

- we split each rule in three parts: a) the element or elements to transform (notice that each rule can transform more than one element); b) the context of the elements to transform (it can be seen as a set of conditions that, being verified, indicate that the rule can be applied); c) the output (specified by a set of functions that will transform the elements according to the chosen representation).
- we assume that there is a set of fixed variables associated with each word. Each variable has the position the word occupies in the text as index. As a result, if two elements are connected (directly or not) they know each other variables, and they can be used to build their formulas.

Moreover, in order to incrementally add new information to our system without having to rewrite more general rules, semantic rules are organised in a subsumption hierarchy. As a result, if a set of rules can be applied, only the rules that do not subsume other rules are triggered.

3.2 Semantic rules

3.2.1 Syntax

Let W be a set of words, C a set of category labels, D a set of dependency labels and _ is used to represent an underspecified value.

Element: $\mathsf{elem}(w, c)$ is an element, where:

- $w \in \{\_\} \cup \mathsf{W}$;
- $c \in \{\_\} \cup \mathsf{C}$.

Arrow: $\mathsf{arrow}(c_1, c_2, d, l)$ is a dependency, and $\mathsf{no\_arrow}(c_1, c_2, d, l)$ a non existing dependency where:

- $c_1, c_2 \in \mathsf{C}$ (c_1 and c_2 are, respectively, the source and the target);
- $d \in \{\_\} \cup \{\mathrm{L}, \mathrm{R}\}$ (d is the dependency orientation: L if it goes from right to left, R from left to right);
- $l \in \{\_\} \cup \mathsf{D}$ (l is a possibly undefined dependency label).

Semantic Rule: $[\mathrm{R}_i]\ \Sigma : \Theta \mapsto \Gamma$ is a semantic rule where:

- Σ is a possibly empty set of elements (the elements to operate on);
- Θ is a possible empty set of existing and non existing dependencies (the rule's context);
- Γ is a set of functions, that vary according to the chosen representation language.

Extra constraints over semantic rules syntax can be found in (Coheur et al., 2003b; Coheur, 2004).

3.2.2 Hierarchy of rules

In the following we define the subsumption relation between semantic rules. This relation establishes the hierarchy of rules and it is based on the subsumption relation between categories. Although we use labels to represent categories, each category is a set of attribute/value pairs organized in a subsumption hierarchy.

Element subsumption: Given $e_1 = \mathsf{elem}(w_1,\ c_1)$ and $e_2 = \mathsf{elem}(w_2,\ c_2)$ from Σ, e_1 subsumes e_2 ($e_1 \sqsubseteq_e e_2$) iff:

- $c_1 \sqsubseteq c_2$;
- $(w_1 \neq \_) \Rightarrow (w_2 = w_1)$.

Dependency subsumption: Given $a_1 = \mathsf{arrow}(c_1,\ c_2,\ d_1,\ l_1)$ and $a_2 = \mathsf{arrow}(c_3,\ c_4,\ d_2,\ l_2)$ from Θ, a_1 subsumes a_2 ($a_1 \sqsubseteq_a a_2$) iff:

- $c_1 \sqsubseteq c_3 \wedge c_2 \sqsubseteq c_4$;
- $(d_1 \neq \_) \Rightarrow (d_2 = d_1)$;
- $(l_1 \neq \_) \Rightarrow (l_2 = l_1)$.

Subsumption of non existing dependencies: Given $a_1 = \mathsf{no\_arrow}(c_1,\ c_2,\ d_1,\ l_1)$ and $a_2 = \mathsf{no\_arrow}(c_3,\ c_4,\ d_2,\ l_2)$ from Θ, a_1 subsumes a_2 ($a_1 \sqsubseteq_a a_2$) iff:

- $c_1 \sqsubseteq c_3 \wedge c_2 \sqsubseteq c_4$;
- $(d_1 \neq \_) \Rightarrow (d_2 = d_1)$;
- $(l_1 \neq \_) \Rightarrow (l_2 = l_1)$.

Rule subsumption: Given two semantic rules $\mathrm{R}_1 = (\Sigma_1,\ \Theta_1,\ \Gamma_1)$ and $\mathrm{R}_2 = (\Sigma_2,\ \Theta_2,\ \Gamma_2)$, R_1 subsumes R_2 ($\mathrm{R}_1 \sqsubseteq_r \mathrm{R}_2$) iff:

- $(\forall\ e_1 \in \Sigma_1)(\exists\ e_2 \in \Sigma_2)\ (e_1 \sqsubseteq_e e_2)$;
- $(\forall\ a_1 \in \Theta_1)(\exists\ a_2 \in \Theta_2)(a_1 \sqsubseteq_a a_2)$.

Finally, if R_1 subsumes R_2, R_2 is said to be more specific than R_1. If both rules can apply, only the most specific rule does so.

3.3 AsdeCopas

AsdeCopas is integrated in a system called Javali (Coheur et al., 2003a), where a module called Ogre (Coheur, 2004) generates a graph, which is AsdeCopas' input. Given the question *Qual a maior praia do Algarve*(*Which is the largest beach in Algarve?*), the following figure shows the graph generated by Ogre:

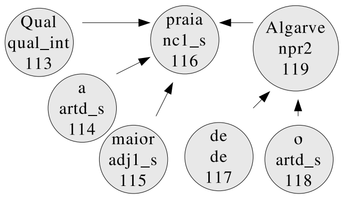

Figure 4: Ogre's output.

Each graph node is a triple, representing: a) a word; b) its associated category; c) its position (in the text). Each graph arrow is also a triple, maintaining information about: a) the position associated with the source node; b) the position associated with the target node; c) the arrow label (possibly undefined)[1].

AsdeCopas is implemented in Prolog. It goes through each graph node and:

- identifies the rules that can be applied;
- chooses the most specific rules;
- triggers the most specific rules.

Then it continuous to the next node. Notice that since rules are self-contained, the way it goes through the graph and the order of rule's application is not relevant, and results remain the same. Notice also, that at each step more than one rule can be triggered.

AsdeCopas is responsible for variable generation. Thus, instead of randomly generating variables, each variable is indexed by the position that the related word occupies in the text. Although apparently naive, this is an important feature of our system which allows different semantic processes to run at different times and results to be merged at the end.

4 Case studies

We present three applications. First we show how AsdeCopas can be used in a disambiguation process. Then we use it to build formulas in MRS (Copestake et al., 2001). Finally, we present an application where AsdeCopas generates logical forms from questions. Quantification is ignored in this last task.

Notice, however, that in order to have a serious evaluation of AsdeCopas capabilities, it needs to be applied to more demanding tasks.

4.1 Disambiguation process

Consider again the quantifier *qualquer*. As we saw, it can take several semantic values. Sometimes the syntactic context allows to limit these

[1]Within our applications, dependencies are unlabelled, and go from dependents to the head. The motivation behind these structures came from the 5P Paradigm.

possibilities. In some situations, one semantic value can be chosen, allowing a full desambiguation.

Let us assume that `all` is an underspecified value (Poesio, 1994) representing all of the semantic values. If no desambiguation takes place, this is the value that will represent this word's semantics. Alternatively, we could opt for a default value. For example, the universal value since it is the most common.

Let us opt for the universal default value. We can write a default rule, as the following:

$[\mathrm{R}_1]$ $\{\mathtt{elem}(\textit{qualquer}, \mathtt{qt})\} : \emptyset$
$\mapsto \{\textit{sem}(\mathtt{qt}) = \mathtt{univ}\}$

Assuming again, as we did in section 2, that on the right of the main verb in the scope of negation, *qualquer* takes the value `indiscriminate` the following rule allows to choose the correct value for *qualquer* in that context:[2]

$[\mathrm{R}_2]$ $\{\mathtt{elem}(\textit{qualquer}, \mathtt{qt})\}$
$: \{\mathtt{arrow}(\mathtt{qt}, \mathtt{n}, \mathrm{L}, \_),$
$\mathtt{arrow}(\mathtt{n}, \mathtt{v}, \mathrm{L}, \_),$
$\mathtt{arrow}(\mathtt{neg}, \mathtt{v}, \mathrm{R}, \_)\}$
$\mapsto \{\textit{sem}(\mathtt{qt}) = \mathtt{indiscriminate}\}$

R_2 is more specific than rule R_1, thus it is applied in these particular conditions. In order to disambiguate, or at least to limit semantic values, other semantic rules would have to be added.

Consider now the Portuguese quantifier *algum*. When it appears on the left side of a noun (`n`), it means "some" (`some`). On the right side it means "none" (`none`), unless it is in the scope of negation. In this particular situation it has an universal value. The following rules allow the right values to be chosen – in this particular situations – for this quantifier (notice that rule R_5 is more specific than rule R_4):

$[\mathrm{R}_3]$ $\{\mathtt{elem}(\textit{algum}, \mathtt{qt})\}$
$: \{\mathtt{arrow}(\mathtt{qt}, \mathtt{n}, \mathrm{R}, \_)\}$
$\mapsto \{\textit{sem}(\mathtt{qt}) = \mathtt{some}\}$

[2]We assume that the object with category `n` arrowing an object with category `v` is the same object with category `n` that receives an arrow from a `qt`. An index is used when we need to distinguish two different objects with the same category.

$[\mathrm{R}_4]$ $\{\mathtt{elem}(\textit{algum}, \mathtt{qt})\}$
$: \{\mathtt{arrow}(\mathtt{qt}, \mathtt{n}, \mathrm{L}, \_)\}$
$\mapsto \{\textit{sem}(\mathtt{qt}) = \mathtt{none}\}$

$[\mathrm{R}_5]$ $\{\mathtt{elem}(\textit{algum}, \mathtt{qt})\}$
$: \{\mathtt{arrow}(\mathtt{qt}, \mathtt{n}, \mathrm{L}, \_)$
$\mathtt{arrow}(\mathtt{n}, \mathtt{v}, \mathrm{L}, \_)$
$\mathtt{arrow}(\mathtt{neg}, \mathtt{v}, \_, \_)\}$
$\mapsto \{\textit{sem}(\mathtt{qt}) = \mathtt{every}\}$[3]

A precise study of the disambiguation of the word *qualquer* can be found in (Coheur, 2003) and (Coheur, 2004), where we try to go as far as possible in the disambiguation process of this word (an some paraphrases of it), by using its syntactic context. Obviously, there are limits to this task, as in some situations information from semantics and pragmatics should also be taken into account to find the correct semantic value.

4.2 Logical forms generation

4.2.1 Minimal Recursion Semantics

Linking syntax with semantics is not an easy task. As Allen says in (Allen, 1995) there seems to be a structural inconsistency between syntactic structure and the structure of the logical form.

We can ease this process by using an adequate representation language. In fact, although the concept is not new (Hobbs, 1983), state of the art frameworks such as (Mollá et al., 2003; Baldridge and Kruijff, 2002) are using flat semantic representations, taht is formulas with no embedded structures (see (Mollá, 2000) for details about flatness), which simplify the syntactic-semantic interface. At the same time, and because it is not reasonable to generate all the possible interpretations of a sentence, many frameworks are using representation languages that leave underspecified semantic interpretations (also an old concept (Woods, 1978)).

MRS (Copestake et al., 2001) uses a flat representation with explicit pointers (called handles) to encode scope effects, corresponding to recursive structures in more conventional formal semantic representations.

We have chosen this language because it has three fundamental characteristics: a) it is a flat language; b) it allows the treatment of quantification; c) it allows underspecification. Un-

[3]Notice, that by choosing the universal value, in the final formula this quantifier will no longer be in the scope of negation.

derspecified MRS structures can be converted into scope-resolved structures that, according to (Copestake et al., 1997), "correspond to those obeyed by a conventionally written bracketed structure".

As an example, MRS represents *Qualquer menino adora algum cão*(*Every boy adores some dog*) in the following underspecified structure (the $=_q$ constraint stands for the equality modulo quantifiers and relates a handle in an argument position to a label (Copestake et al., 2001)):

```
top p[4]
h1:every(x, r1, n), h3:menino(x),
r1 =_q h3, h7:cão(y),
h5:some(y, r5, m), r5 =_q h7,
h4:adora(e, x, y)
```

where `h1` outscopes `h3` and `h5` outscopes `h7`.

Then, by means of a set of constraints, such that an MRS structure must be a tree, there should be a unique top-level handle, etc., the following readings are obtained:

```
p=h1 (wide scope "every")
h1:every(x, h3, h5), h3:menino(x),
h5:some(y, h7, h4), h7:cão(y),
h4:adora(e, x, y)

p=h5 (wide scope "some"),
h5:some(y, h7, h1), h7:cão(y),
h1:every(x, h3, h4), h3:menino(x),
h4:adora(e, x, y)
```

In the next section we will show how to reach these formulas.

4.2.2 Toy example

We will show how to reach an underspecified MRS representation for constructions as *Qualquer*$_{67}$ *menino*$_{68}$ *adora*$_{69}$ *a*$_{70}$ *Maria*$_{71}$[5] and *Qualquer*$_{678}$ *menino*$_{679}$ *adora*$_{680}$ *algum*$_{681}$ *cão*$_{682}$. Notice that, for expository reasons, we are simplifying the process. Actual rules use fine grained categories for quantifiers, and scope restrictions are imposed differently (Coheur, 2004).

In order to perform this task we use the following functions:

[4]`p` is the variable over the top.

[5]*Every boy adores Maria*

- *sem* returns a (default) predicate
  ex: *sem*(*Maria*) = `Maria`[6];
- *var* returns a variable
  ex: *var*(*Maria*) = $\mathtt{x}_{71}$;
- *handle* returns a variable for an handle
  ex: *handle*(*Maria*) = $\mathtt{h}_{71}$;
- *restrictor* returns a variable for a restrictor
  ex: *restrictor*(*Maria*) = $\mathtt{r}_{71}$;
- *scope* returns a scope variable
  ex: *scope*(*Maria*) = $\mathtt{s}_{71}$.

The following rule applies to nouns, either common nouns (nc) or proper nouns (npr), everytime it finds one (because the arrow set is empty).

[R$_1$]{`elem`(_, n)}
: $\emptyset$
$\mapsto$ {*handle*(n): *sem*(n)(*var*(n)}

If only this rule is defined, the first sentence is translated into:

$\mathtt{h}_{68}$`:menino(`$\mathtt{x}_{68}$`)`
$\mathtt{h}_{71}$`:Maria(`$\mathtt{x}_{71}$`)`

and the second sentence into:

$\mathtt{h}_{679}$`:menino(`$\mathtt{x}_{679}$`)`
$\mathtt{h}_{682}$`:cão(`$\mathtt{x}_{682}$`)`

Nonetheless, $\mathtt{h}_{71}$`:Maria(`$\mathtt{x}_{71}$`)` is not the representation we want for *Maria*. Instead we use the predicate `NAME`. Thus, we define R$_2$, subsumed by R$_1$ (because n $\sqsubseteq$ npr), and consequently more specific.

[R$_2$]{`elem`(_, npr)}
: $\emptyset$
$\mapsto$ {*handle*(npr):`NAME`(*var*(npr), *sem*(npr))}

Rule R$_2$ is triggered instead of R$_1$ and we obtain for the first sentence

$\mathtt{h}_{71}$`:NAME(`$\mathtt{x}_{71}$`, Maria)`

instead of

$\mathtt{h}_{71}$`:Maria(`$\mathtt{x}_{71}$`)`.

[6]Although these values can be obtained in a lexicon, in this application they are generated from sentence words.

Notice that a new rule needs to be defined for the situations where the npr arrows an nc and not a v, since we want to translate *mãe*$_{80}$[7] *Maria*$_{81}$ into `mãe(`x_{80}`), NAME(`x_{80}`, Maria)` and not into `mãe(`x_{80}`), NAME(`x_{81}`, Maria)`. In order to do this, we need only to add a rule for npr (like the previous rule) to be applied when a npr arrows an nc. This rule, being more specific than rule R$_2$, is applied in this particular situation. As the npr is connected with the nc, it "knows" its variable, which can be used is the associated formula.

The next rule is applied to a verb (v) when the verb has an n arrowing from left (typically the subject) and an n arrowing from right (typically the direct object), and no preposition arrows these nouns.

[R$_3$]{`elem(_, v)`}
: {`arrow(`n_i`, v, R, _)`,
`arrow(`n_j`, v, L, _)`,
`no_arrow(prep, `n_i`, R)`,
`no_arrow(prep, `n_j`, R)`}
$\mapsto$ {*handle*(v):*sem*(v)(*var*(v),*var*(n_i), *var*(n_j))}

As a result, in the first sentence, *adora* is translated into:

h_{69}`:adora(`x_{69}`, `x_{68}`, `x_{71}`)`

and, in the second one, it is translated into:

h_{680}`:adora(`x_{680}`, `x_{679}`, `x_{682}`)`.

Notice that, at this point, although we don't have rules for all the elements within the example sentences, we already have a partial representation.

Consider now, a generic rule for quantifiers (qt):

[R$_4$] {`elem(_,qt)`}
: {`arrow(qt, nc, _, _)`}
$\mapsto$ {*handle*(qt): *sem*(qt)(*var*(nc), *restrictor*(qt), *scope*(qt)), *restrictor*(qt) $=_q$ *handle*(nc)}

Now, the results depend on previous processing: if the disambiguation task described in the previous section was performed, *sem*(qualquer) = `every` and *sem*(algum) = `some`. Otherwise, underspecified values are used.

[7] *mother.*

Let us consider that the disambiguation stage took place before. Thus, this rule adds to the first sentence:

h_{67}`: every(`x_{68}`, `r_{67}`, `s_{67}`)`,
r_{67} $=_q$ h_{68}

and to the second sentence:

h_{678}`: every(`x_{679}`, `r_{678}`, `s_{678}`)`,
r_{678} $=_q$ h_{679}

and

h_{681}`: some(`x_{682}`, `r_{681}`, `s_{681}`)`,
r_{681} $=_q$ h_{682}.

Notice that we reach the underspecified formula from 4.2.1 for the first sentence.

We will conclude now this example. It should be clear that additional rules could impose extra constraints to the formula, avoiding spurious ambiguities.

4.3 Question interpretation

From system Edite we inherited a corpus of 680 questions about tourist resources and we made a preliminar evaluation over 30 questions. There was no pre-processing of these questions, no compound terms were detected, no mistakes were corrected.

Let "correct representation" be a set of formulas representing a question, where the exact number of expected predicates are produced, and variables are in the correct places.

Let "system representation" be the set of formulas that the system suggests as the question representation. Each question can have more than one "system representation". Moreover, a correct "system representation" is a "system representation" that is equal to the "correct representation".

A general evaluation (of the whole system) results in a precision of 0,19 (number of correct "system representations"/number of total "system representations") and a recall of 0,77 (number of correct "system representations"/30). If we eliminate two particularly bad results (one associated 42 "system representations" to a question and the other 21), we have a precision of 0,37.

The low precision results from previous stages, as several graphs are associated with each question. In fact, with the actual set of semantic rules only one representation is associated with each graph.

Nevertheless, the analysis is not over. The majority of "system representations" produced by AsdeCopas are just incomplete and result from unknown words. For example, the statement *Quais os roteiros pedestres sinalizados em Lisboa?*(*Which are the signalised footways in Lisbon?*), originated the following formula, where `AM` is the predicate for adjectival modification and `?` indicates the focus of the question:

```
?x759
roteiros(x759)
AM(x759, x760), pedestres(x760)
em(x759, x763)
NAME(x763, Lisboa)
```

As the word *sinalizados* was not recognised, the "system representation" is not correct, because a predicate associated with this word is missing. Nevertheless, most of the information contained in the question is retrieved.

Within AsdeCopas framework a special effort was made with paraphrastic relations. As an example, both phrases *Quais os hotéis com piscina?*(*Which are the hotels with a swimming pool?*) and *Em que hotéis há piscina?*(*In which hotels is there a swimming pool?*), result in the following formulas:

```
?x22
hotéis(x22)
com(x22, x24)
piscina(x24)
```

Note that in order to reach this result, we had just to look into the particular syntactic conditions that make verb *haver* (*to have*) behave as the preposition `com` (*with*).

5 Conclusions

We presented AsdeCopas, a syntax-semantics interface based on hierarchically organized semantic rules. AsdeCopas is integrated in a system called Javali and it has been applied to several tasks. Apart from some adjustments, AsdeCopas should be able to process any dependency structure.

In the near future, we will have to study coordination properly. We also indent to extend our work to English.

6 Acknowledgements

We are greatful for the corrections of Diamantino Caseiro, David Matos and Sérgio Paulo. We also acknowledge Ricardo Ribeiro and Rui Chaves. Finally, we thank for the useful comments of this paper reviewers. As usual, the responsability for the contents of this paper lies with the authors alone.

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.

James Allen. 1995. *Natural Language Understanding (second edition)*. The Benjamin Cummings Publishing Company, Inc.

Jason Baldridge and Geert-Jan M. Kruijff. 2002. Coupling CCG and hybrid logic dependency semantics. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 319–326.

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

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

Luísa Coheur, Fernando Batista, and Joana Paulo. 2003a. JaVaLI!: understanding real questions. In *EUROLAN 2003 student workshop: Applied Natural Language Processing, possible applications for the Semantic Web*, Bucharest, Romania, July.

Luísa Coheur, Nuno Mamede, and Gabriel G. Bés. 2003b. ASdeCopas: a syntactic-semantic interface. In Fernando Moura Pires and Salvador Abreu, editors, *Progress in Artificial Intelligence: 11th Portuguese Conference on Artificial Intelligence, EPIA 2003*, volume 2902 / 2003 of *Lecture Notes in Artificial Inteligence*, Beja, Portugal, Dezembro. Springer-Verlag.

Luísa Coheur. 2003. A situação do "qualquer" em qualquer situação. Technical Report RT/004/03-CDIL, L^{2}F-Laboratório de Sistemas de Língua Falada, Inesc-id, Lisboa, Portugal, Março.

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.

Ann Copestake, Dan Flickinger, and Ivan A. Sag. 1997. Minimal recursion semantics: An introduction (draft).

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2001. Minimal recursion semantics: An introduction. *L&C*, 1(3):1–47.

Luísa Marques da Silva, Nuno Mamede, and David Matos. 1997. Edite - um sistema de acesso a uma base de dados em linguagem natural. In *Workshop sobre taggers para o português*, pages 20–33, Lisboa, Portugal. Instituto de Linguística Teórica e Computacional.

Luísa Marques da Silva. 1997. Edite, um sistema de acesso a base de dados em linguagem natural, análise morfológica, sintáctica e semântica. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal.

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

Jerry R. Hobbs. 1983. An improper treatment of quantification in ordinary english. In *21st Annual Meeting of the Association for Computational Linguistics (ACL).*

Telmo Móia. 1992. *Aspectos da Semântica do Operador Qualquer (Cadernos de Semântica nº 5).* Faculdade de Letras da Universidade de Lisboa.

Diego Mollá. 2000. Ontologically promiscuous flat logical forms for NLP. In *IWCS-4*, Tilburg, The Netherlands.

Diego Mollá and Ben Hutchinson. 2002. Dependency-based semantic interpretation for answer extraction. In *Proceedings of the Australasian NLP Workshop (ANLP'02)*, Canberra.

Diego Mollá, Rolf Schwitter, Fabio Rinaldi, James Dowdall, and Michael Hess. 2003. Extrans: Extracting answers from technical texts. *IEEE Intelligent Systems*, 18(4), July/August.

M. Poesio. 1994. Ambiguity, underspecification and discourse interpretation. In R. A. Muskens H. Bunt and G. Rentier (eds.), editors, *Proceedings of the First International Workshop on Computational Semantics*, pages 151–160. "ITK, Tilburg University".

Paulo Reis, J. Matias, and Nuno Mamede. 1997. Edite - a natural language interface to databases: a new dimension for an old approach. In *Proceedings of the Fourth International Conference on Information and Communication Technology in Tourism (ENTER'97)*, pages 317–326, Edinburgh, Escócia. Springer-Verlag, Berlin, Germany.

W. A. Woods. 1978. Semantics and quantification in natural language question answering. In M. Yovitz, editor, *Advance in Computers*, volume 17. New York: Academic Press. Reprinted in *Readings in Natural Language Processing*, edited by B. Grosz, K. Jones and B. Webber and published by Morgan Kaufmann Publishers, Inc. in 1986.