Meaning⇔Text theory and Dependency Tree Semantics: an account of Underspecification

Leonardo Lesmo, Livio Robaldo & Serena Villata

Dipartimento di Informatica - Università di Torino
C.so Svizzera 185 – 10149 Torino - Italy
{lesmo, robaldo, villata}@di.unito.it

Abstract

This paper presents an approach to semantic interpretation for MTT that accounts for ambiguities of quantifier scoping. Dependency Tree Semantics (DTS) is described as a target formalism that introduces in the SemS a mechanism for dealing with quantifiers. DTS is based on underspecification, i.e. all ambiguous interpretations of a quantified sentence are represented in a single structure, where the scope relations are not given. The underspecified representations can be extended by adding dependency arcs that express the scope ordering when enough information is available for the disambiguation. Since a single underspecified structure covers all readings of a sentence, the mapping from the SSyntS to the SemS is heavily simplified. In order to provide some evidence about this, the paper includes the definition of the syntax-semantics mapping for a small fragment of English.

Keywords

Semantics, syntax-semantics interface, quantifiers, underspecification

1 Introduction

The Meaning⇔Text Theory (MTT) aims at representing explicitly the m:n correspondence between Texts/Sounds and Meanings of Natural Language utterances. In standard MTT (see (Mel’čuk, 1988), (Mel’čuk, 2001) and (Kahane, 2003a)) this correspondence is carried out along a bidirectional chain of levels of linguistic description. Rules are defined to relate adjacent levels. Since MTT is strongly grounded in the lexicon (“It is 100% lexicon-based” (Mel’čuk, 2004a)), great attention has been paid to the syntactic and semantic “slots” describing the relations existing among the participants in a situation. In particular, the links between Semantemes in the Semantic-Structures (SemS) are mapped to relations at the Deep-Syntactic Structures (DSynt) level according to rules aimed at “ensuring the complete preservation of meaning” (Mel’čuk, 2004b). These rules, for instance, block the syntactic realization of some semantic actants under some specific conditions, or, viceversa, force this
realization for some given verbs or in given contexts. Also, the difference between actants and modifiers (or circumstantialis) has been deeply investigated, as a way to characterize the semantic and syntactic behavior of specific lexical units. Comparatively less effort has been devoted to the analysis of linguistic elements that are less concerned with the semantic-syntactic behavior of content words, in particular quantifiers. As Kahane puts it “the scope of a quantifier is not directly encoded in standard MTT semantic representations, where quantifiers are monoactantial semantemes whose argument is the quantified semanteme. Therefore, nothing in the semantic graph of All men are looking for a cat indicates whether or not a cat is in the scope of all.” (Kahane, 2003b). In this paper, we present an approach to semantic representation that aims at simplifying the mapping between the Semantic and the Syntactic levels with respect to the representation of quantifiers. The paper is focused on the interpretation of sentences, so that the mapping that will be presented is transductive (Kahane, 2003b), with a syntactic representation given as input and its semantic representation obtained as result. The syntactic structures correspond to the surface syntactic level of MTT, so the syntactic/semantic interface describes a direct mapping from surface syntax to semantics. According to MTT, there is an intermediate level (i.e. the Deep Syntactic Level) acting as a bridge between surface syntax and semantics. However, it has been argued that “the correspondence from the semantic level to the surface-syntactic level must be direct and the deep-syntactic structure is the witness of this correspondence” (Kahane, 2003a). In this view, the DSyntS is a “witness” in the sense that it represents a “trace” of how the SSyntS is obtained from the SemS: it encodes the steps that were applied to obtain the SSyntS, exactly as the Derivation Tree in Tree Adjoining Grammar does (Rambow & Joshi, 1997). Consequently, the syntactic dependency structures that will be presented in the next section must be viewed as corresponding to SSyntS. With respect to the output of the transducer, the semantic representation (Dependency Tree Semantics) has been designed with a twofold goal: to enable a uniform treatment of ambiguities of quantifier scopes and to be as close as possible to the syntactic structure. Consequently, we paid most attention to quantified structures and to the interaction among them. With respect to the other fundamental component of a semantic representation (i.e. the predicate-argument structures), the MTT approach already provides most useful insights. In this paper, we describe the features of the semantic representation, assuming a standard MTT treatment of actants and circumstantialis, and focusing on the interpretation of quantifiers. It must be observed that we address the problem of obtaining a representation covering “all” possible readings of a sentence. It has been argued that in some languages, some of the readings are banned because of syntactic constraints. We believe that in many cases the syntactic constraints are somewhat “fuzzy”, in the sense that they embody preferences more than strict rules. In any case, these constraint are assumed to be applied “after” the set of possibilities has been identified. This does not affect the correctness of the final result. The paper is organized as follows: in the next section, we present the syntactic dependency structures, in the third section, we introduce the main features of the semantic representation, while the fourth one is devoted to the syntax/semantics interface. A section on related work and the conclusions close the paper.

2 Syntactic Structures

The Dependency Structures we adopt as input of the transducer are being used at the University of Torino in a number of projects, including the development of a Treebank for Italian (http://www.di.unito.it/~tutreeb/) and the implementation of a rule-based dependency parser (Lesmo & Lombardo, 2002). The dependency arcs are labelled according to a scheme
that encodes the surface relations. The scheme is based on a distinction between Functional and Non-functional dependents. The latter are dependents not having domain-based semantic import, and are classified into Aux (auxiliaries), Coordinator (arcs related with conjunctions), Contin (continuations, in idioms), Separator (punctuation marks), Visitors (e.g. in raising structures), Interjections, and some particles void of semantic contents (as the Italian "accorger-si" – remark, where the -si reflexive pronoun is lexicalized into a pseudo-reflexive verb). The Function class is split into Arguments and Modifiers, corresponding to the standard distinction between syntactic actants and circumstancials (see (Mel’čuk, 2004b)). The modifiers can be Rmod (restrictive modifiers) or Appositions. See Fig.1.

Figure 1: Syntactic dependency tree associated with the sentence “Dove posso comprare due biglietti per l’opera Il Flauto Magico?” (Where can [I] buy two tickets for the opera The Magic Flute?)

With respect to the dependency structures usually adopted in MTT, we must note that determiners (and quantifiers) are the head of nominal subtrees, i.e. they govern the associated noun. This is in agreement with the solution proposed in Word Grammar (Hudson, 1990). Fig.1 also includes two traces; empty nodes include links (called Coref) to their referents; these links, however, are not standard dependency arcs, so that the Projectivity condition is respected. The figure also includes an example of a compound name (Il Flauto Magico, i.e. The Magic Flute), treated as an idiom (contin+denom) and acting as a denomination apposition (noun-apposition+denom) of the noun “opera”. More details on the labelling scheme can be found in (Bosco & Lombardo, 2003). It has also been applied to English and its relations with the Paninian scheme are currently under study.

3 The semantic representation

The semantic representation of the sentences is given according to Dependency Tree Semantics (DTS). DTS was introduced in (Lesmo & Robaldo, 2005); the most complete description is (Robaldo, 2007). DTS aims at providing a semantic structure strictly related to the dependency-based syntactic structure (from which its name) and at accounting for the fact that, in syntactic trees, the scope of quantifiers is not explicitly given. In order to achieve these goals, DTS must abandon the standard linear representation of logical formulae; as in
SSemS, it adopts a graph-based representation. Since the scope of quantifiers is fundamental in enabling correct inferences, the linear order that encodes the scope relations must be replaced by some other device. The device adopted in DTS is akin to Skolem functions: the “dependencies” among quantifiers are encoded via explicit arcs. Skolem theory states that, when an $\exists$ quantifier occurs inside the scope of $n$ $\forall$ quantifiers, the existentially quantified variable can be replaced by a function of the $n$ universally quantified ones. So, $\forall x_1 \forall x_2 \ldots \forall x_n \exists y \ p(x_1, x_2, \ldots, x_n, y)$ and $\forall x_1 \forall x_2 \ldots \forall x_n \ p(x_1, x_2, \ldots, x_n, f(x_1, x_2, \ldots, x_n))$, are equi-satisfiable, or, more precisely, “there exists some function $f$” (a second-order quantification) such that the two formulae are equi-satisfiable; i.e. $y$ depends in some way (according to $f$) on the $x_i$. In DTS, this dependency is encoded explicitly via suitable arcs (see below). An example of a representation given according to DTS is in Fig. 2. The left part of Fig. 2 reports the syntactic structure of the sentence:

(1) Every man heard a mysterious sound

The right-hand side shows the associated DTS, which is given in three parts:
- The predicate-argument structure, which is depicted as a graph corresponding with the SemS (this graph is called Flat Dependency Graph – FDG)
- The restr function takes as argument the variables in the semantic tree and its values are subtrees of that tree. This function encodes a link between syntax and semantics, by stating which part of the tree acts as a restriction for each variable
- The quant function, specifying the quantifiers associated with each variable

Figure 2: Syntactic and semantic representation for the sentence “Every man heard a mysterious sound”

The representation in Fig. 2 is underspecified, in the sense that it corresponds to two different readings of the sentence, that are represented, in standard predicate logic, as:

\[
\begin{align*}
(1) \quad a. & \exists y [ \text{sound'}(y) \land \text{mysterious'}(y) \land \forall x [\text{man'}(x) \rightarrow \text{hear'}(x,y) ]] \\
& \forall x [\text{man'}(x) \rightarrow \exists y [\text{sound'}(y) \land \text{mysterious'}(y) \land \text{hear'}(x,y) ]] 
\end{align*}
\]

In (1a), the $\forall$ quantifier is included in the scope of $\exists$, while the opposite holds for (1b). Therefore, (1a) says that a single sound has been heard by all considered men. In contrast, (1b) says that for each man there is a potentially different sound that was heard by him. It may be argued that scope disambiguation is fundamental in logical reasoning, but it is less important when the main goal is paraphrasing (as in MTT). However, it is rather clear that a sentence as “There is a mysterious sound heard by every man” is a reasonable paraphrase for (1a), but not for (1b). In other words, paraphrasing could lead to expressions that, because of syntactic constraints, are “less ambiguous” than the original one. So, it seems that also for
paraphrasing a precise specification of the scope relations is important\(^1\). As said above, the
disambiguation of a basic DTS representation (see Fig.2) is obtained by introducing new arcs
in the graph. These arcs are called \textit{SemDep} arcs, and explicitly specify the scope order of the
quantifiers. The two readings of sentence (1) are shown in Fig.3.

\begin{figure}[h]
\centering
\begin{minipage}[b]{0.3\textwidth}
\centering
\begin{tikzpicture}[scale=0.7]
  \node[shape=circle,draw] (CTX) at (0,0) {CTX};
  \node[shape=circle,draw] (man) at (-2,-2) {man'};
  \node[shape=circle,draw] (sound) at (-1,-3) {sound'};
  \node[shape=circle,draw] (mysterious) at (0,-3) {mysterious'};
  \node[shape=circle,draw] (hear) at (-2,-4) {hear'};
  \draw[->] (CTX) to node [left] {1} (man);
  \draw[->] (CTX) to node [left] {1} (sound);
  \draw[->] (CTX) to node [left] {1} (mysterious);
  \draw[->] (CTX) to node [above] {2} (hear);
  \draw[->] (man) to node [right] {1} (sound);
  \draw[->] (man) to node [right] {1} (mysterious);
  \draw[->] (sound) to node [right] {1} (mysterious);
  \draw[->] (hear) to node [right] {1} (man);
\end{tikzpicture}
\end{minipage}
\hspace{2cm}
\begin{minipage}[b]{0.3\textwidth}
\centering
\begin{tikzpicture}[scale=0.7]
  \node[shape=circle,draw] (CTX) at (0,0) {CTX};
  \node[shape=circle,draw] (man) at (-2,-2) {man'};
  \node[shape=circle,draw] (sound) at (-1,-3) {sound'};
  \node[shape=circle,draw] (mysterious) at (0,-3) {mysterious'};
  \node[shape=circle,draw] (hear) at (-2,-4) {hear'};
  \draw[->] (CTX) to node [left] {1} (man);
  \draw[->] (CTX) to node [left] {1} (sound);
  \draw[->] (CTX) to node [left] {1} (mysterious);
  \draw[->] (CTX) to node [above] {2} (hear);
  \draw[->] (man) to node [right] {1} (sound);
  \draw[->] (man) to node [right] {1} (mysterious);
  \draw[->] (sound) to node [right] {1} (mysterious);
  \draw[->] (hear) to node [right] {1} (man);
\end{tikzpicture}
\end{minipage}
\hspace{2cm}
\begin{minipage}[b]{0.3\textwidth}
\centering
\begin{tikzpicture}[scale=0.7]
  \node[shape=rectangle,draw] (x) at (0,0) {x} node[above] {quant(x): \(\forall\)};
  \node[shape=rectangle,draw] (y) at (1,0) {y} node[above] {quant(y): \(\exists\)};
  \node[shape=circle,draw] (CTX) at (-2,0) {CTX};
  \node[shape=circle,draw] (man) at (-3,-2) {man'};
  \node[shape=circle,draw] (sound) at (-2,-3) {sound'};
  \node[shape=circle,draw] (mysterious) at (-1,-3) {mysterious'};
  \draw[->] (CTX) to node [left] {1} (man);
  \draw[->] (CTX) to node [left] {1} (sound);
  \draw[->] (CTX) to node [left] {1} (mysterious);
  \draw[->] (CTX) to node [above] {2} (x);
  \draw[->] (CTX) to node [above] {2} (y);
  \draw[->] (man) to node [right] {1} (sound);
  \draw[->] (man) to node [right] {1} (mysterious);
  \draw[->] (sound) to node [right] {1} (mysterious);
  \draw[->] (x) to node [right] {1} (man);
  \draw[->] (y) to node [right] {1} (sound);
\end{tikzpicture}
\end{minipage}
\end{figure}

\begin{itemize}
\item[(c)]
\item[(a)]
\item[(b)]
\item[(d)]
\end{itemize}

\textbf{Figure 3: Disambiguated DTS for the sentence “Every man heard a mysterious sound”}

It can be noted that a new node has been included in the representation: \textit{CTX}. It refers to “the
context”: in case a \textit{SemDep} arc enters it, the quantifier of the associated variable has widest
scope. Fig.3(a) corresponds to the reading where the universal quantifier has wider scope.
This is expressed by the fact that the \(y\) variable “depends on” \(x\), i.e. that the value of \(y\) can be
determined only when \(x\) is given (\(y\) is a function of \(x\), in Skolem terms). In Fig.3(b), we have a
rather different situation: in this case, both variables “depend on” \textit{CTX}. This is due to the
constraint that universal quantifiers always depend on the context (i.e. “every man” cannot
depend on another variable). But, in this case, the same holds for \(y\), so that it is also
determined only on the basis of the context, i.e. it is a constant. This apparently
dishomogeneous situation actually is one of the major advantages of DTS. In fact it enables it
to represent the so-called “branching quantifier readings”, that are usually hard for standard
(e.g. predicate logic) approaches. In sentence (2), there are two (numerical) quantifiers.

\begin{enumerate}
\item[2] Two students studied three theorems
\end{enumerate}

In predicate logic, we can get the following formulae:

\begin{enumerate}
\item[2a] \(2x \left[ \text{student}'(x) \land 3y \left[ \text{theorem}'(y) \land \text{study}'(x,y) \right] \right] \)
\item[2b] \(3y \left[ \text{theorem}'(y) \land 2x \left[ \text{student}'(y) \land \text{study}'(x,y) \right] \right] \)
\end{enumerate}

In (2a) each student can study different theorems, so that there are exactly two students and
up to six involved theorems. Conversely, in (2b), there exist exactly three theorems, and each
of them can be studied by two different students. However, there is a third reading where
exactly two students and three theorems are involved. Because of the linear order imposed by
the syntax of logical formulae, this reading is hard to represent: either \(2x\) precedes \(3y\) or
viceversa. In DTS, if we make both \(x\) and \(y\) depend on \textit{CTX}, we get the desired result.
Another advantage of DTS is that what appears in Fig.2 needs only additions (of suitable
\textit{SemDep} arcs) in order to obtain the scoped version: no reshuffling of the graph is needed, as
is required in \textit{underspecified} representations (Hobbs & Shieber, 1987) (Reyle, 1993) (Bos,

\footnote{Note that the underspecified representation refers only to semantics. This paper does not address the
important topic of devising a single graph representation for different syntactic structures.}
Leonardo Lesmo, Livio Robaldo & Serena Villata

In fact, the definition of \textit{restr} and \textit{quant} functions (Fig.3(c-d)) is not affected by the disambiguation of the FDG. As in (Polguère, 1997), the semantemes associated with determiners express a quantification of a first actant over a second one, the former being included in the latter. Contrary to Polguère, however, the first actant (called \textit{restriction} as in standard Generalized Quantifier perspective) is a subnetwork and the second one, not explicitly represented, consists in the subnetwork of all functor-actant dependencies where the semanteme occurs, hence, also the ones in the restriction.

4 The syntax-semantics interface

As the name “Dependency Tree Semantics” suggests, the syntax-semantic interface relates DTS with a Dependency Grammar (see (Kahane & Lareau 2005) for an approach based on Unification). In particular, the G$\rightarrow$DTS interface allows to associate a Dependency Tree in G with a Flat Dependency Graph (FDG) in DTS. This interface is compositional, i.e. a Dependency Tree is related to only one FDG. The G$\rightarrow$DTS interface is a generalization of XDG (Debusmann, 2006). The main difference is that the former relates structures with different sets of nodes. In fact, the nodes of a Dependency Tree in G are words like \textit{study}, \textit{every}, while the nodes in a FDG are predicates like \textit{study'}, \textit{xJohn}, or discourse referents like \textit{x}, \textit{y}, \textit{z}. The mapping between nodes in G and nodes in FDG is expressed via two functions \textit{Sem} and \textit{SVar} (\textit{Lex} and \textit{LVar} implements the inverse mapping), that are defined below. In the very simple grammar G considered here for the sake of example, the domain objects (\textit{DomG}) are words of English vocabulary classified into seven Parts Of Speech (POS): \textit{IV} (intransitive verbs), \textit{TV} (transitive verbs), \textit{PN} (proper names), \textit{CN} (common nouns), \textit{PREP} (prepositions), \textit{DET} (determiners), and \textit{RP} (relative pronouns). The grammatical functions are subject (\textit{verb-subj}), object (\textit{verb-obj}), prepositional modifier (\textit{prep-rmod}), verbal modifier (for relative clauses: \textit{verb-rmod}), argument of a preposition (\textit{prep-arg}), and argument of a determiner (\textit{det-arg}). The semantic actants are identified, as usual, by numeric labels. The domain objects of a FDG (\textit{DomDTS}) are a set of semantic objects that can be predicates or discourse referents (\textit{pred} and \textit{D}, respectively). A proper name corresponds to a constant (in \textit{const}), associated with a predicate \textit{ια}, where \textit{α}$\in$\textit{const} is the constant denoted by the proper name and \textit{ι} is a standard operator that, applied to a constant, builds a predicate that is true for that constant and false otherwise. Since \textit{α} is a predicate, it also belongs to \textit{pred}. \textit{Sem} relates verbs, common nouns, and other content words in G to a predicate in \textit{pred}. \textit{SVar} relates determiners, proper names, and relative pronouns in G to a discourse referent in \textit{D}. Having defined \textit{Sem} and \textit{SVar}, it is possible to state how to obtain a FDG from a Dependency Tree of G. The mapping is defined by a set of if-then rules between “pieces” of structures, called Link$_{G,FDG}$. Two example rules are reported in Fig.4. They specify how the subjects of verbs and the arguments of determiners are interpreted: the rule in Fig.4(a) asserts that the subject of an intransitive (transitive) verb corresponds to the first argument of the unary (binary) predicate associated to the verb by \textit{Sem}. The arguments of these predicates are the discourse referents associated to the dependent of the verb by \textit{SVar}. Analogous rules constrain the semantic realization of the direct object and the argument of a preposition.
The rule in Fig.4(b) deals with arguments of determiners. Here, the direction of the arc is reversed, since, in G, the det-arg link leads from the determiner to the noun whereas the predicate-argument relation links the predicate associated with the noun to the discourse referent (corresponding to the quantified variable) associated with the determiner. Adjuncts are dealt with via rules with an additional level of complexity; in particular, prepositional and verbal modifiers link a common noun (associated with a predicate $p_1$) to a preposition or a verb (associated with another predicate $p_2$); clearly, $p_1$ and $p_2$ have to be applied to the same discourse referent. To this end, we introduce in the rules a new variable $d$, referring to this common discourse referent; however, contrary to the variables $v$ and $u$, in the rules there is no constraint on $d$’s counterpart (i.e. on $LVar(d)$). These rules are shown in Fig.5.

As a final example, we show the rules for dealing with proper names and relative pronouns. As stated above, a proper name $p$ has to be mapped onto two different nodes, e.g. $\iota\alpha$ and $x$, where $\alpha \in \text{const}$ is the constant associated with $p$ and $x$ is the discourse referent corresponding to the individual $p$ denotes. Furthermore, $x$ has to be the argument of the unary predicate $\iota\alpha$. On the contrary, a relative pronoun has to correspond to a discourse referent already introduced by a determiner. To handle this case, we introduce a function $\text{Ref}$ that associates a relative pronoun with its referent. These two rules are shown in Fig.6.

The last ingredient needed to build the underspecified semantic representation is a criterion to set the value of the function $\text{restr}(x)$, for each variable $x$ in the $\text{FDG}$. $\text{restr}(x)$ is simply set to the subgraph of all predicate-argument relations $R(x_1, \ldots, x_n)$ such that:

- a) $R(x_1, \ldots, x_n)$ arises from the subtree having the determiner associated with $x$ as root.

---

1 The value of $\text{quant}$ depends only on the lexical meaning of the associated NL quantifier.
b) $x$ is one of the $x_1, ..., x_n$

This seems to be consistent with the ideas lying behind the architecture of a Dependency Tree; in fact, in a dependency relation between a word head and a word dependent, the latter plays a role of “completion” of the former, in the sense that it circumscribes/restricts the head meaning by acting as its parameter. However, it has been pointed out that not only the dependent is involved in this completion, but the whole subtree having the dependent as root.

A final example shows how the different rules interact in the transduction process. In Fig.7, on the left, we show the syntactic tree of “All students studied two theorems of the book”. Most steps of the transduction are trivial (as the application of the rules in Fig.4 to the substructures “studied –verb-subj–> all”, “all –det-arg–> students”, “two –det-arg–> theorems”, “the –det-arg–> book”). As stated above, “studied –verb-obj–> two”, is handled via a rule as the one for verb-subj. One tricky point concerns the prepositional modifier: “theorems –prep-rmod–> of –prep-arg–> the”. Here, the applied rule is the left one in Fig.5. The variable $d$ appearing in the rule is unified with the discourse object $y$, whose existence is due to the application of the det-arg rule (Fig.4(b)) to ”two theorems”. The application of the prep-rmod rule leads to the inclusion of the “of” –1–> $y$ arc. In other words, in the case of prep-rmod, the semantic effect is to add further arcs to already existing discourse objects. This sounds rather natural, since the involved discourse objects had to be created because of their presence in a connected structure (e.g. as actants of verbs) and the modifiers just add them some more specifications. Of course, the latter are also represented as participants in other relations (in the example, the of’ relations, which could be expanded in “being part of a book”).

![Figure 7: Transduction of the sentence “All students studied two theorems of the book”](image)

### 5 Related work

In standard MTT, the scope of quantifiers is not directly encoded (Kahane, 2003b) and, to our knowledge, just two relevant solutions have been proposed to this end. The first one (Dymetman & Copperman, 1996) distinguishes between two types of semantic structures: U-forms and S-forms. The latter is a “scoped” version of the former. For instance, the meaning of (1) is represented via the U-form in Fig.8(a). The nodes of the graphs can be seen as semantemes while the labels (det, 1, 2, ..., -1, -2,...) specify, with respect to the source node, either a determiner (det) or a predicate argument position (positive labels mark complements while negative ones mark modifiers). This representation does not establish any scope order between the two quantifiers. The U-form in Fig.8(a) can then be converted into one of the S-forms in Fig.8(b–c). They include the same predicate-arguments connections of the U-form, but define a scoping between the quantifiers since the order on the nodes in the tree is relevant. In particular, it is assumed that each node has scope over its right siblings.
The S-forms are model-theoretically interpreted via a translation into equivalent λ-formulae. The λ-formulae associated with the two S-forms in Fig.8 are shown below them.

\[
\exists (\lambda h_1.\text{sound}(h) \land \text{myst}(h), \\
\forall (\lambda h_2.\text{man}(l), \lambda l_1 \lambda l_2.\text{hear}(l_1, l_2)))
\]
\[
\exists (\lambda h_2.\text{man}(l), \exists (\lambda h_1.\text{sound}(h) \land \\
\text{myst}(h), \lambda l_1 \lambda l_2.\text{hear}(l_1, l_2)))
\]

Figure 8: Representation of “Every man heard a mysterious sound” in Dymetman and Copperman’s approach

A second relevant approach is the one of (Polguère, 1997). In his proposal, quantifiers are represented as biactantal functors. The first actant of a functor is a semanteme, while the second actant is a subnetwork containing the first actant. Quantifiers express the quantification of the first actant in the second actant. For instance, in Polguère’s view, the two interpretations of (1) could be represented as in Fig.9. The SemS in Fig.9(a) describes the reading in which a single sound is heard by every man, while in the SemS in Fig.9(b) each involved man has heard a potentially different sound.

\[
\forall (\lambda h_2.\text{man}(l), \exists (\lambda h_1.\text{sound}(h) \land \\
\text{myst}(h), \lambda l_1 \lambda l_2.\text{hear}(l_1, l_2)))
\]

Figure 9: Representation of “Every man heard a mysterious sound” in Polguère’s approach

6 Conclusions

In this paper, we have presented Dependency Tree Semantics (DTS) as a tool for providing a formal semantics to quantifiers in MTT. DTS is a semantic formalism putting at disposal underspecified representations that are very close to the syntactic structures in SSynS. These structures, called Flat Dependency Graph (FDG), can be disambiguated by adding to the FDG suitable arcs (SemDep arcs) specifying the scope of quantifiers. DTS has a formal model-based semantics, which could not be presented in the paper because of space constraints (see (Robaldo, 2007)) and a very natural syntax-to-semantics interface, which was developed on the basis of Extensible Dependency Grammar framework. The interface is briefly presented in the paper, in order to give an overall picture of the process leading from the SSynS to a proper semantic representation. The proposal described in the paper mainly concerns the management of quantifier scope, a topic often disregarded in MTT approaches to semantics.
The integration of this proposal with other approaches for handling actants and circumstantials could be a significant step to a wide-coverage semantic interpretation in MTT.

Bibliography


