Towards a large-scale formal semantic lexicon for text processing*

Johan Bos
Department of Computer Science
University of Rome "La Sapienza", Italy
bos@di.uniroma1.it

Abstract

We define a first-order fragment of Discourse Representation Theory (DRT) and present a syntax-semantics interface based on Combinatory Categorial Grammar (CCG) (with five basic categories N, NP, S, PP and T) for a large fragment of English. In order to incorporate a neo-Davidsonian view of event semantics in a compositional DRT framework we argue that the *method of continuation* is the best approach to deal with event modifiers. Within this theoretical framework, the lexical categories of CCGbank were associated with the formal semantic representations of our DRT variant, reaching high coverage of which practical text processing systems can benifit.

1 Introduction

Formal approaches in semantics have long been restricted to small or medium-sized fragments of natural language grammars. In this article we present a large-scale, formally specified lexicon backed up by a model-theoretic semantics. The aim of the lexicon is to provide wide-coverage parsers with the possibility to produce interpretable structures in a robust yet principled way.

The semantic lexicon that we present is situated in Discourse Representation Theory (Kamp & Reyle, 1993), and it follows to a great extent the theory as formulated by its orginator Hans Kamp and colleagues. However, it deviates on certain points, as it comprises:

- a neo-Davidsonian view on representing event structures;
- a syntax-semantics interface based on categorial grammar and type-theory;
- a DRS language compatible with first-order logic.

In a neo-Davidsonian take on event semantics events are first-order entities and characterised by one-place predicate symbols. An inventory of thematic roles encoded as two-place relations between the events and the subcategorised arguments or modifiers completes this picture. We choose this way

* Published in: *Proc. GSCL 2009* TODO . Seite **??–??**.

of representing events because it yields a more consistent analysis of event structure. As inventory of thematic roles we take VerbNet (Kipper *et al.*, 2008).

As a preliminary to semantics, we need syntax. The syntax-semantics interface illustrated here is based on a categorial grammar, namely Combinatory Categorial Grammar, or CCG for short (Steedman, 2001). A categorial grammar lends itself extremely well for this task because it is lexically driven and has only few "grammar" rules, and not less because of its type-transparency principle, which says that each syntactic type (a grammar category) corresponds to a unique semantic type. The existence of CCGbank (Hockenmaier, 2003), a large collection of CCG derivations for a newswire corpus, and the availability of robust parsers trained on it (Clark & Curran, 2004), make CCG further a practically motivated choice.

The choice of first-order logic for the semantic interpretation of text is a restriction in terms of expressiveness. However, it opens the way to implement logical reasoning in practical systems by including automated deduction tools such as theorem provers and finite model builders for inference tasks such as consistency and informativeness checking (Blackburn & Bos, 2005).

In the scope of this article we are primarily interested in defining proper semantic representations for lexical categories. Hence resolving scope ambiguities, word sense disambiguation, thematic role labelling, and anaphora resolution are tasks outside the scope of this article. They are, of course, essential in a complete system for computational semantics, but these are tasks orthogonal to the objectives of producing a formally grounded semantic lexicon. Instead, the challenges in the context of this article are providing well-formed, interpretable, lexical semantic representations for a broad variety of linguistic phenomena, and doing this on a large-scale, producing a lexicon suited for wide-coverage, high precision natural language processing grammars for open-domain text analysis.

2 Discourse Representation Theory (DRT)

DRT is a theory of natural language meaning, and was originally designed by Kamp to study the relationship between indefinite noun phrases and anaphoric pronouns as well as temporal relations (Kamp, 1981). Since its publication in the early 1980s, DRT has grown in depth and coverage, establishing itself as a well-documented formal theory of meaning, dealing with a stunning number of semantic phenomena ranging from pronouns, abstract anaphora, presupposition, tense and aspect, propositional attitudes, ellipsis, to plurals (Kamp & Reyle, 1993, Klein, 1987, van der Sandt, 1992, Asher, 1993, van Eijck & Kamp, 1997, Geurts, 1999, Kadmon, 2001).

DRT can be divided into three components. The central component is a formal language defining **Discourse Representation Structures** (DRSs), the meaning representations for texts. The second component deals with the **semantic interpretation** of DRSs. The third component constitutes an algorithm that systematically maps natural language text into DRSs, the **syntax-semantics interface**. Let's consider these components in more detail in our version of DRT.

One of the main principles of the theory is that a DRS can play both the role of semantic *content*, and the role of discourse *context* (van Eijck & Kamp, 1997). The content gives us the precise model-theoretic meaning of a natural language expression, and the context it sets up aids in the interpretation of subsequent anaphoric expressions occuring in the discourse. A key ingredient of a DRS is the *discourse referent*, a concept going back to at least the work of Karttunen (1976). A discourse referent is an explicit formal object storing individuals introduced by the discourse, along

with its properties, for future reference. The recursive structure of DRSs determines which discourse referents are available for anaphoric binding.

Semantic interpretation of DRSs is carried out by translation to first-order logic. The DRS language employed in our large-scale lexicon is nearly identical to that formulated in Kamp & Reyle (1993). It is, on the one hand, more restrictive, leaving out the so-called duplex conditions because they do not all permit a translation to first-order logic. Our DRS language forms, on the other hand, an extension, as it includes a number of modal operators on DRSs not found in Kamp & Reyle (1993) nor van Eijck & Kamp (1997). This DRS language is known to have a translation to ordinary first-order formulas. Examples of such translations are given in Kamp & Reyle (1993), Muskens (1996), and Blackburn *et al.* (2001), disregarding the modal operators. A translation incorporating the modal operators is given by Bos (2004). We won't give the translation in all its detail here, as it is not the major concern of this article, and interested readers are referred to the articles cited above.

Various techniques have been proposed to map natural language text into DRS: the top-down algorithm (Kamp & Reyle, 1993), DRS-threading (Johnson & Klein, 1986), or compositional methods (Zeevat, 1991, Muskens, 1996, Kuschert, 1999, van Eijck & Kamp, 1997, de Groote, 2006). We will follow the latter tradition, because it enables a principle way of constructing meaning permitting broad coverage — it also fits very well with the grammar formalism of our choice, CCG. Hence, in addition to the formal language of DRSs, we need a "glue language" for **partial DRSs** — we will define one using machinery borrowed from type theory. This will give us a formal handle on the construction of DRSs in a bottom-up fashion, using function application and β-conversion to reduce complex expressions into simpler ones.

Let's turn to the definition of partial DRSs.[1] The basic semantic types in our inventory are e (individuals) and t (truth value)[2]. The set of all types is recursively defined in the usual way: if τ_1 and τ_2 are types, then so is $\langle \tau_1, \tau_2 \rangle$, and nothing except the basic types or what can be constructed via the recursive rule are types. Expressions of type e are either discourse referents, or variables. Expressions of type t are either basic DRSs, DRSs composed with the merge (;), or DRSs formed by function application (@):

<exp$_e$> ::= <ref> | <var$_e$>

$$\text{<exp}_t\text{>} ::= \boxed{\begin{array}{c}\text{<ref>}* \\ \hline \text{<condition>}*\end{array}} \mid (\text{<exp}_t\text{>};\text{<exp}_t\text{>}) \mid (\text{<exp}_{\langle\alpha,t\rangle}\text{>} \ @ \ \text{<exp}_\alpha\text{>})$$

Following the conventions in the DRT literature, we will visualise DRSs in their usual box-like format. As the definition above shows, basic DRSs consist of two parts: a set of discourse referents, and a set of conditions. The discourse referents can be seen as a record of topics mentioned in a sentence or text. The conditions tell us how the discourse referents relate to each other, and put further semantic constraints on their interpretation. We distinguish between basic and complex conditions. The basic conditions express properties of discourse referents or relations between them:

<condition> ::= <basic> | <complex>

<basic> ::= <sym$_1$>(<exp$_e$>) | <sym$_2$>(<exp$_e$>,<exp$_e$>) | <exp$_e$>=<exp$_e$>
| card(<exp$_e$>)=<num> | named(<exp$_e$>,<sym$_0$>)

[1] We employ Backus-Naur form in the definitions following. In using this notation, non-terminal symbols are enclosed in angle brackets, choices are marked by vertical bars, and the asterix suffix denotes zero or more repeating items.

[2] Type t corresponds to truth value in static logic, however in a dynamic logic like DRT one might want to read it as a state transition, following van Eijck & Kamp (1997).

Here <sym$_n$> denotes an n-place predicate symbol, and <num> a cardinal number. Most nouns and adjectives introduce a one-place relation; prepositions, modifiers, and thematic roles introduce two-place relations. (In our neo-Davidsonian DRS language we don't need ternary or higher-place relations.) The cardinality condition is used for counting quantifiers, the naming condition for proper names. The equality condition explicitly states that two discourse referent denote the same individual.

Now we turn to complex conditions. For convenience, we split them into unary and binary complex conditions. The unary complex conditions have one DRS as argument and cover negation, the modal operators expressing *possibility* and *necessity*, and a "hybrid" condition connecting a discourse referent with a DRS. The binary conditions have two DRSs as arguments and form implicational, disjunctive, and interrogative conditions:

<complex> ::= <unary> | <binary>

<unary> ::= $\neg$<exp$_t$> | $\Box$<exp$_t$> | $\Diamond$<exp$_t$> | <ref>:<exp$_t$>

<binary> ::= <exp$_t$>$\Rightarrow$<exp$_t$> | <exp$_t$>$\vee$<exp$_t$> | <exp$_t$> ? <exp$_t$>

The unary complex conditions are mostly activated by negation particles or modal adverbs. The hybrid condition is used for the interpretation of verbs expressing propositional content. The binary complex conditions are triggered by conditional statements, certain determiners, coordination, and questions.

Finally, we turn to expressions with complex types. Here we have three possibilities: variables, λ-abstraction, and function application:

<exp$_{\langle\alpha,\beta\rangle}$> ::= <var$_{\langle\alpha,\beta\rangle}$> | λ<var$_\alpha$>.<exp$_\beta$> | (<exp$_{\langle\gamma,\langle\alpha,\beta\rangle\rangle}$>@<exp$_\gamma$>)

To graphically distinguish between the various types of variables we will make use of different (indexed) letters denoting different types (Table 1). This table also illustrates the language of partial DRSs with some first examples.

Note that variables can range over all possibly types, except type t. This restriction permits us to use the standard definition of β-conversion to avoid accidental capturing of free variables by forcing the functor to undergo the process of α-conversion. Applying a functor expression to an argument expression of type t could result in breaking existing bindings, because DRSs can bind variables outside their syntactic scope. Since we don't have variables ranging over type t in our language of partial DRSs, this can and will never happen. In practical terms, the syntax-semantics interface is not affected by this limitation, and therefore the price to pay for this restriction of expressiveness is low.

3 Combinatory Categorial Grammar (CCG)

Every semantic analysis presupposes a syntactic analysis that tells us how meaning representations of smaller expressions can be combined into meaning representations of larger expressions. The theory of syntax that we adopt is CCG, a member of the family of categorial grammars (Steedman, 2001). In a categorial grammar, all constituents, both lexical and composed ones, are associated with a syntactic category. Categories are either *basic categories* or *functor categories*. Functor categories indicate the nature of their arguments and their directionality: whether they appear on the left or on the right. In CCG, directionality of arguments is indicated within the functor category by slashes: a

Table 1: Examples of (partial) DRSs

Type	Variable	Expression	Example
e	x, x_1, x_2, ... e, e_1, e_2, ...		
t		x e manager(x) smoke(e) theme(e,x)	a manager smoked
$\langle e, t \rangle$	p, p_1, p_2, ...	λx. [; company(x)]	company
$\langle\langle e, t \rangle, t \rangle$	n, n_1, n_2, ...	λp.([x ; bank(x)] ;(p@x))	a bank

forward slash / tells us that an argument is to be found on its immediate right; a backward slash \ that it is to be found on its immediate left.

The inventory of basic categories in a CCG grammar usually comprises S, N, PP, and NP, denoting categories of type sentence, noun, prepositional phrase, and noun phrase, respectively (Steedman, 2001). Functor categories are recursively defined over categories: If α is a category and β is a category, then so are (α/β) and ($\alpha\backslash\beta$).[3] Note that, in theory, there is no limit to the number of categories that we are able to generate. In practice however, given the way natural language works, the number of arguments in a functor category rarely succeeds four.

Consider for instance S\NP, a functor category looking for a category of type NP on its immediate left, yielding a category of type S. This category, of course, corresponds to what traditionally would be called an intransitive verb or verb phrase in a grammar for English. Other examples of functor categories are N/N, a pre-nominative adjective, and (S\NP)/NP, a transitive verb.

These simple examples demonstrate a crucial concept of a categorial grammar: functor categories encode the subcategorisation information directly and straightforwardly. All such syntactic dependencies, local as well as long-range, are specified in the categories corresponding to lexical phrases. This process of "lexicalisation" is a trademark property of categorial grammars and manifests itself in a lexicon exhibiting a large variety of categories complemented with a small set of grammar rules — the essence of a categorial grammar.

The most simplest of categorial grammar, *pure categorial grammar*, has just two combinatory rules: forward and backward application. CCG can be viewed as a generalisation of categorial grammar, and the initial C it is blessed with is due to the variety of combinatory rules that it adds to the grammar of pure categorial grammar. The complete collection of binary combinatory rules in CCG consists of the application rules, the rules for composition and their crossing variants (including the generalised versions), and the substitution rules (not covered in this article). In addition, CCG comes with rules for type raising. Some variations of CCG have special rules for dealing with coordination.

[3] Outermost brackets of a functor category are usually suppressed.

The Application Rules

Forward application ($>$ in Steedman's notation), and backward application ($<$), are the two basic combinatory rules of classic categorial grammar. Below we will give the schemata of these rules, and make use of the colon to associate the category with its semantic interpretation. Following the partial-DRS language defined in the previous section, we use @ to denote function application.

$$\frac{\text{X/Y: } \phi \qquad \text{Y: } \psi}{\text{X: } (\phi @ \psi)} > \qquad\qquad \frac{\text{Y: } \phi \qquad \text{X}\backslash\text{Y: } \psi}{\text{X: } (\psi @ \phi)} <$$

Here X and Y are variables ranging over CCG categories, ϕ and ψ are partial DRSs. The $>$ rule can be read as follows: given an expression of category X/Y with interpretation ϕ, followed by an expression of category Y with interpretation ψ, we can deduce a new category X, with interpretation ϕ applied to ψ. Along the same lines, the $<$ rule cam be paraphrased as follows: given the category X$\backslash$Y with interpretation ψ, preceeded by a category Y with interpretation ψ, we can deduce a new category X, with interpretation ψ applied to ϕ.

The Composition Rules

Here we have forward and backward composition, rules that were originally introduced to deal with various cases of natural language coordination. Steedman refers to these rules by $>$**B** and $<$**B**, blessed after the Bluebird from Raymond Smullyan's tale of the "logical forest" and its feathered friends.

$$\frac{\text{X/Y: } \phi \qquad \text{Y/Z: } \psi}{\text{X/Z: } \lambda x.(\phi @ (\psi @ x))} >\mathbf{B} \qquad\qquad \frac{\text{Y}\backslash\text{Z: } \phi \qquad \text{X}\backslash\text{Y: } \psi}{\text{X}\backslash\text{Z: } \lambda x.(\psi @ (\phi @ x))} <\mathbf{B}$$

The $>$**B** rule can be paraphrased as follows: given an expression of category X/Y, we're looking for an expression of category Y on our right, but find an expression of category Y/Z instead, then we can deduce X/Z (because, after all, we did encounter Y, if we can promise to find a Z later as well). A similar explanation can be given for the $<$**B** rule, reversing the direction of subcategorisation. Semantically, something interesting is happening here. Since we haven't find Z yet, we need to postpone its semantic application. What we do is introduce the variable x and apply it to the interpretation of Y/Z (or Y$\backslash$Z in the $<$**B** rule), and then abstract over it again using the λ-operator.

Both composition rules have so-called "crossing" variants: backward crossing composition ($<\mathbf{B}_x$) and forward crossing composition ($>\mathbf{B}_x$). Backward crossing occurs in cases of adjunctal modification in English. Forward crossing is not needed for an English grammar, but supports languages with freer word order such as Italian. The crossing rules are specified as follows:

$$\frac{\text{X/Y: } \phi \qquad \text{Y}\backslash\text{Z: } \psi}{\text{X}\backslash\text{Z: } \lambda x.(\phi @ (\psi @ x))} >\mathbf{B}_x \qquad\qquad \frac{\text{Y/Z: } \phi \qquad \text{X}\backslash\text{Y: } \psi}{\text{X/Z: } \lambda x.(\psi @ (\phi @ x))} <\mathbf{B}_x$$

The Generalised Composition Rules

The composition rules also have *generalised* variants, in order to guarantee a larger variety of modification types. Steedman introduces a technical device to abstract over categories, the \$, a placeholder for a bounded number of directional arguments (Steedman, 2001).

$$\frac{\text{X/Y: } \phi \qquad \text{(Y/Z)\$: } \psi}{\text{(X/Z)\$: } \lambda\vec{x}.(\phi@(\psi@\vec{x}))} >\mathbf{B}\$ \qquad \frac{\text{(Y\Z)\$: } \phi \qquad \text{X\Y: } \psi}{\text{(X\Z)\$: } \lambda\vec{x}.(\psi@(\phi@\vec{x}))} <\mathbf{B}\$$$

$$\frac{\text{X/Y: } \phi \qquad \text{(Y\Z)\$: } \psi}{\text{(X\Z)\$: } \lambda\vec{x}.(\phi@(\psi@\vec{x}))} >\mathbf{B}_x\$ \qquad \frac{\text{(Y/Z)\$: } \phi \qquad \text{X\Y: } \psi}{\text{(X/Z)\$: } \lambda\vec{x}.(\psi@(\phi@\vec{x}))} <\mathbf{B}_x\$$$

Here we introduce a vectorised variable as short-hand notation for the number of abstractions and applications respectively required. This number, of course, depends on the number of arguments of the functor category that is involved.

Type Raising

CCG also comes with two type raising rules, forward and backward type raising. Steedman calls these $>\mathbf{T}$ and $<\mathbf{T}$, respectively, after Smullyan's Thrush. They are unary rules, and effectively what they do is change an argument category into a functor category. The standard CCG example illustrating the need for type raising is non-constituent coordination such as right-node raising, a phenomenon that can be accounted for in CCG by combining type raising with forward composition. The type raising rules are defined as follows:

$$\frac{\text{X: } \phi}{\text{Y/(Y\X): } \lambda x.(x@\phi)} >\mathbf{T} \qquad \frac{\text{X: } \phi}{\text{Y\(Y/X): } \lambda x.(x@\phi)} <\mathbf{T}$$

Type-raising is an extremely powerful mechanism, because it can generate an infinitely large number of new categories. Usually, in practical grammars, restrictions are put on the use of this rule and the categories it can apply to.

4 Building a Formal Semantic Lexicon

The method we follow to construct a large-scale semantic lexicon has two parts: a theoretical part, defining a mapping between the base categories of CCG and semantic types; and a practical part, assigning to each lexical category a suitable partial DRS of the right type, guided by CCG's type transparency principle (Steedman, 2001). As we have seen from the previous section, the combinatory rules of CCG are equipped with a direct semantic interpretation, completing the syntax-semantics interface. Recall that the main basic categories in CCG are N (noun), S (sentence), NP (noun phrase) and PP (prepositional phrase). In addition we will take on board the basic category T (text) and motivate this addition below. Our main task here is to map these basic categories to the two basic semantic types we have at our disposal: e (entities), and t (truth value).

The Category N

The CCG category N is assigned a semantic type $\langle e, t\rangle$, a type corresponding to *properties*. This makes sense, as what nouns are essentially doing is expressing kinds of properties. This decision allows us to view some first examples of associating lexical items with λ-DRS. So let's consider the word *squirrel* of category N, and the adjective *red* of category N/N and their semantic representations:

N: $\langle e, t\rangle$: λx. $\boxed{\begin{array}{|c|}\hline\\ \hline \text{squirrel(x)}\\ \hline\end{array}}$ N/N: $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$: λp.λx.($\boxed{\begin{array}{|c|}\hline\\ \hline \text{red(x)}\\ \hline\end{array}}$;(p@x))

The Category PP

We also assign the type $\langle e, t\rangle$ to the category PP. It is perhaps confusing and misleading to connect two different syntactic categories with the same semantic type (because it seems to obscure CCG's principle of type transparancy (Steedman, 2001)). However, from a semantic perspective, both N and PP denote properties, and it seems just to assign them both $\langle e, t\rangle$ (see the example for *at a table* below). One could attempt to make a semantic distinction by sorting entities into individuals and eventualities, because in CCG a PP usually plays the role of a subcategorised argument of a verb. But this wouldn't lead us anywhere, as the PP could, in principle, also be an argument of a relational noun as illustrated below for *wife*:

PP: $\langle e, t\rangle$: $\lambda x_1.$ $\begin{array}{|c|}\hline x_2 \\ \hline \text{table}(x_2) \\ \text{at}(x_1,x_2) \\ \hline\end{array}$

N/PP: $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$: $\lambda p.\lambda x_1.($ $\begin{array}{|c|}\hline x_2 \\ \hline \text{person}(x_1) \\ \text{wife}(x_2) \\ \text{role}(x_1,x_2) \\ \hline\end{array}$;(p@x_2))

The Category NP

At first thought it seems to make sense to associate the type e with the category NP, as for example Steedman (2001) does. After all, a noun phrase denotes an entity. However, this is not what we propose, because it would require a type-raised analysis in the lexicon for determiners to get proper meaning representations for quantifiers. In CCGBank (Hockenmaier, 2003), on which our practical implementation is based, determiners such as *every* are categorised as NP/N rather than their type-raised variants (T/(T\NP))/N and (T\(T/NP))/N, which would permit us to assign the type e to NP and still give a proper generalised quantifier interpretation. The motivation of CCGBank to refrain from doing this is obviously of practical nature: it would increase the number of categories drastically. Nevertheless, by associating the category NP with the type $\langle\langle e, t\rangle, t\rangle$ we are able to yield the required interpretation for determiners.

Essentially, what we are proposing is a uniform type-raised analysis for NPs, denoting functions from properties to truth values. This is, of course, an idea that goes back to Montague's formalisation for a fragment of English grammar (Montague, 1973), illustrated by the following example for *someone*:

NP: $\langle\langle e, t\rangle, t\rangle$: $\lambda p.($ $\begin{array}{|c|}\hline x \\ \hline \text{person}(x) \\ \hline\end{array}$;(p@x))

The Category S

Now we turn to the category for sentences, S. Normally, one would associate the type t to S: after all, sentences correspond to propositions and therefore denote truth values. Yet, the semantics of events that we will assign to verbal structures, following Davidson, requires us to connect the category S with the type $\langle\langle e, t\rangle, t\rangle$. As this is slightly unconventional, we will motivate this choice in what follows.

The neo-Davidsonian approach that we are following yields DRSs with discourse referents denoting event-like entities. For instance, the DRS for the sentence *a manager smoked* would be one similar as the one shown in Table 1. This would be a fine DRS. But what if we continue the sentence, with a modifier, as in *a manager smoked in a bar*, or with a sequence of modifiers, as in *a manager*

smoked in a bar yesterday? As we wouldn't have any λ-bound variables at our disposal, it is impossible to connect the modifiers *in a bar* and *yesterday* to the correct discourse referent for the event (e). In CCG, such modifiers would typically be of category S\S or S/S (i.e. sentence modifiers) and (S\NP)\(S\NP) or (S\NP)/(S\NP) (i.e. verb phrase modifiers). It is not difficult to see that associating a DRS to category S would lead us astray from the principles of compositional semantics, when trying to maintain a neo-Davidsonian first-order modelling of event structures. We would need an ad-hoc mechanism to ensure the correct binding of the event discourse referent. Several "tricks" could be performed here. Event discourse referents could always have the same name (say e_0), and modifiers would be represented with a free variable of the same name. This would require a modification of the definition of closed expressions and variable renaming — it would also exclude a DRS with two different events discourse referents. Another possibility is to treat modifiers as anaphoric, linking to the event discourse referent closest in proximity. This would establish a dichotomy in the analysis of modifiers, and would moreover require modification of the β-conversion procedure too, as variables ranging over type t would be introduced. No further comments needed — such ad-hoc methods aren't welcome in any systematic syntax-semantic interface.

There are two basic directions to surmount this problem, while maintaining a compositional system. The first is to abstract over the event variable, and introduce the discourse referent when the parse of the sentence is completed. We call this the *method of delay* and it would yield the type $\langle e, t\rangle$ for the S category. The second approach is to introduce the discourse referent for the event in the lexicon, but reserve a place for any modifier that comes along later in the parse. This option, which we dub the *method of continuation*, would yield the type $\langle\langle e, t\rangle, t\rangle$ for S. We show the partial DRSs for both options:

λe. | x | manager(x) smoke(e) agent(e,x) |

λp.(| x e | manager(x) smoke(e) agent(e,x) | ;(p@e))

Is there a practical difference between the two options? Yes, there is. The delayed approach (shown on the left) will always introduce the discourse referent on the outermost level of DRS. This will give the incorrect prediction for sentences with scopal operators such as quantifiers in subject position. The continuation approach (shown on the right) doesn't suffer from this limitation, as the discourse referent is already introduced. Moreover, the continuation approach also gives us control over where in the DRS modifiers are situated which the delay approach doesn't. This motivates us to favour the continuation approach. To illustrate the impact of the method of continuation on the lexical categories, consider the partial DRS for the intransitive verb *smoked*:

(S[dcl]\NP): $\langle\langle\langle e, t\rangle, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$: $\lambda n_1.\lambda p_2.(n_1@\lambda x_3.($ | e_4 | smoke(e_4) agent(e_4,x_3) | $;(p_2@e_4)))$

The Category T

The category T (for text) corresponds to the semantic type t and as a consequence to an ordinary DRS (or an expression that can be reduced to a DRS) which can be translated to first-order logic. It is not usually a lexical category, but typically introduced by punctuation symbols that signal the

end of the sentence. For instance, the full stop maps to a CCG category that turns a sentence into a text: T\S. This makes sense from a linguistic point of view, because a full stop (or other punctuation symbols such as exclamation and question marks) signals that the sentence is finished and no more sentence of verb phrase modifiers will be encountered.

5 Practical Results

Our inventory of lexical categories is based on those found in CCGbank (Hockenmaier, 2003) and used by the C&C parser for CCG (Clark & Curran, 2004). CCGbank is a version of the Penn Treebank (Marcus *et al.*, 1993) and consists of a set of CCG derivations covering 25 sections of texts taken from the Wall Street Journal, comprising a total of over one million tagged words. CCGBank contains nearly 49,000 sentences annotated with a total of 1,286 different CCG categories, of which 847 appear more than once.

Given the theoretical foundations presented in the previous sections, we manually encoded the majority of the lexical categories found in CCGbank with partial DRSs (categories not used by the C&C parser, and some extremely rare categories, were not taken into account). Even though there is a one-to-one mapping between the CCG categories and semantic types — and this must be the case to ensure the semantic composition process proceeds without type clashes — the actual ingredients of a partial DRSs can differ even within the scope of a single CCG category. A case in point is the lexical category N/N can correspond to an adjective, a superlative, a cardinal expression, or even common nouns and proper names (in compound expressions). In the latter two cases the lexical entry introduces a new discourse referent, in the former cases it does not. To deal with these differences we also took into account the part of speech assigned by the C&C parser to a token to determine an appropriate partial DRS.

This system for semantic interpretation was implemented and when used in combination with the C&C parser achieves a coverage of more than 99% on re-parsed Wall Street Journal texts, and similar figures on newswire text. With *coverage* we mean here the percentage of sentences for which a well-formed DRS could be produced (not necessarily a DRS that is semantically adequate). The robustness of the overall framework for deep text processing is good enough to make it possible for practical tasks with non-restricted domains, such as open-domain question answering (Bos *et al.*, 2007) and recognising textual entailment (Bos & Markert, 2005).

Summing up: formal semantics isn't a paper-and-pencil game anymore, nor is it limited to implementations of toy fragments of natural language. It has genuinely matured to a level useful for applications in the real world.

References

Asher, N. (1993). *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers.

Blackburn, P. & Bos, J. (2005). *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI.

Blackburn, P., Bos, J., Kohlhase, M., & de Nivelle, H. (2001). Inference and Computational Semantics. In H. Bunt, R. Muskens, & E. Thijsse, editors, *Computing Meaning Vol.2*, pages 11–28. Kluwer.

Bos, J. (2004). Computational Semantics in Discourse: Underspecification, Resolution, and Inference. *Journal of Logic, Language and Information*, **13**(2), 139–157.

Bos, J. & Markert, K. (2005). Recognising textual entailment with logical inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 628–635.

Bos, J., Guzzetti, E., & Curran, J. R. (2007). The Pronto QA System at TREC 2007: Harvesting Hyponyms, Using Nominalisation Patterns, and Computing Answer Cardinality. In E. Voorhees & L. P. Buckland, editors, *Proceeding of the Sixteenth Text RETrieval Conference, TREC-2007*, pages 726–732, Gaithersburg, MD.

Clark, S. & Curran, J. (2004). Parsing the WSJ using CCG and Log-Linear Models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*, Barcelona, Spain.

de Groote, P. (2006). Towards a montagovian account of dynamics. In *Proceedings of Semantics and Linguistic Theory XVI (SALT 16)*.

Geurts, B. (1999). *Presuppositions and Pronouns*. Elsevier, London.

Hockenmaier, J. (2003). *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Johnson, M. & Klein, E. (1986). Discourse, anaphora and parsing. In *11th International Conference on Computational Linguistics. Proceedings of Coling '86*, pages 669–675, University of Bonn.

Kadmon, N. (2001). *Formal Pragmatics*. Blackwell.

Kamp, H. (1981). A Theory of Truth and Semantic Representation. In J. Groenendijk, T. M. Janssen, & M. Stokhof, editors, *Formal Methods in the Study of Language*, pages 277–322. Mathematical Centre, Amsterdam, Amsterdam.

Kamp, H. & Reyle, U. (1993). *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht.

Karttunen, L. (1976). Discourse referents. In J. McCawley, editor, *Syntax and Semantics 7: Notes from the Linguistic Underground*, pages 363–385. Academic Press, New York.

Kipper, K., Korhonen, A., Ryant, N., & Palmer, M. (2008). A large-scale classification of english verbs. *Language Resources and Evaluation*, **42**(1), 21–40.

Klein, E. (1987). VP Ellipsis in DR Theory. *Studies in Discourse Representation Theory and the Theory of Generalised Quantifiers*.

Kuschert, S. (1999). *Dynamic Meaning and Accommodation*. Ph.D. thesis, Universität des Saarlandes.

Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, **19**(2), 313–330.

Montague, R. (1973). The proper treatment of quantification in ordinary English. In J. Hintikka, J. Moravcsik, & P. Suppes, editors, *Approaches to Natural Language*, pages 221–242. Reidel, Dordrecht.

Muskens, R. (1996). Combining Montague Semantics and Discourse Representation. *Linguistics and Philosophy*, **19**, 143–186.

Steedman, M. (2001). *The Syntactic Process*. The MIT Press.

van der Sandt, R. (1992). Presupposition Projection as Anaphora Resolution. *Journal of Semantics*, **9**, 333–377.

van Eijck, J. & Kamp, H. (1997). Representing Discourse in Context. In J. van Benthem & A. ter Meulen, editors, *Handbook of Logic and Language*, pages 179–240. Elsevier, MIT.

Zeevat, H. W. (1991). *Aspects of Discourse Semantics and Unification Grammar*. Ph.D. thesis, University of Amsterdam.