Towards Wide-Coverage Semantic Interpretation

Johan Bos
University of Edinburgh

1 Introduction

Wide-coverage and robust NLP techniques always seemed to go hand in hand with shallow analyses. This was certainly true a couple of years ago, but the state-of-the-art in stochastic approaches has advanced considerably and nowadays there are sophisticated parsers available achieving high coverage and producing accurate syntactic analyses. It seems we have finally reached a stage in NLP where we can apply well known techniques of formal and computational semantics to a larger scale, and get a detailed semantic analysis from a wide-coverage parser. A proof of concept of this idea was demonstrated in [BCS$^+$04], with a coverage of over 95% on newspaper texts. In this paper we discuss the further developments in this work, generating semantic representations for sentences or small texts, showing how we can calculate background knowledge required for reasoning, and performing inferences using state-of-the-art theorem provers and model builders.

The semantic representation language that we will use is a first-order language, arguing that given the current state of automated deduction, any language with more expressive power (such as second or higher-order logic) cannot be used efficiently to perform inference tasks. There are however highly sophisticated inference tools for first-order logic available which we will use in our work.

Despite the tradition in formal semantics to use higher-order logics, first-order logic is able to cover a (perhaps surprisingly) large variety of interesting natural language phenomena. The language we are going to adopt is developed in Discourse Representation Theory (DRT), closed under a translation to first-order logic, and is described in Section 2. The choice for DRT is motivated by its impressive theoretical coverage of linguistic phenomena [KR93, VdS92].

Next, of course, we need a grammar formalism suitable for computational semantics (i.e. one that is able to produce fine-grained syntactic analyses).

We will use Combinatory Categorial Grammar (CCG) for this task, mainly for three reasons: CCG handles a variety of long-distance dependencies and coordination cases, there are fast parsers available achieving high coverage and accuracy, and, finally, as CCG is a lexicalised theory of grammar, it is an ideal framework to implement a compositional semantics. We will discuss CCG and the syntax-semantics interface in Section 3.

However, just building semantic representations is not enough. If one is willing to exploit them in their full power, that is, applying some kind of logical inference, there is more required than just the semantics stemming from the words: background knowledge. In Section 4 we show how to calculate a part of this background knowledge (arguable the most important part), by automatically generating ontological knowledge in the form of first-order axioms using the WordNet lexical database.

2 The Semantic Formalism

2.1 The Core DRS Language

The semantic representations that we build comprise the basic DRS language (as formulated in [KR93]) using a neo-Davidsonian analysis of events, together with a theory of presuppositional expressions includin anaphora. DRSs come with a model-theoretical interpretation, but the interpretation that we will use is one via a translation to first-order logic, as formulated in [Bos04]. This is advantageous from a practical and computational perspective, because it opens the door to use automated theorem provers for first-order logic to implement inference tasks [BBKdN01].

The syntax of DRSs and DRS-conditions are defined by simultaneous recursion, with respect to a set of first-order variables and a vocabulary describing the predicate symbols and their respective arities. This is done using the following clauses:

1. If $\{\mathrm{x}_1 \dots \mathrm{x}_n\}$ is a finite set of variables, $\{\gamma_1 \dots \gamma_m\}$ a finite set of DRS-conditions, then the ordered pair $\langle\{\mathrm{x}_1 \dots \mathrm{x}_n\}, \{\gamma_1 \dots \gamma_m\}\rangle$ is a DRS;
2. If R is a relation symbol for an n-place predicate and $\mathrm{x}_1 \dots \mathrm{x}_n$ are variable s then $R(\mathrm{x}_1, \dots, \mathrm{x}_n)$ is a DRS-condition;
3. If x_1 and x_2 are variables, then $\mathrm{x}_1 = \mathrm{x}_2$ and $\mathrm{x}_1 \neq \mathrm{x}_2$ is a DRS-condition;
4. If B is a DRS, then $\neg$B, $\Box$B, $\Diamond$B are DRS-conditions;
5. If B_1 and B_2 are DRSs, then $\mathrm{B}_1 \vee \mathrm{B}_2$ and $\mathrm{B}_1 \Rightarrow \mathrm{B}_2$ are DRS-conditions;
6. If x is a variable and B a DRS, then x:B is a DRS-condition.

Clause 1 defines DRSs in the standard way. Given a DRS B=$\langle D, C\rangle$, D is called the domain of B, C are the conditions of B, and members of D are called B's discourse referents. The basic conditions (clause 2–3) are defined just as in standard DRT. Clause 4 introduces negation and the modal operators, and Clause 5 disjunction and implication. Clause 6 is non-standard—it introduces a modal operator that explicitly associates variables ranging over possible worlds with DRSs. As in [Bos04], we will use it to analyse constructions with sentential complements and discourse relations.

2.2 Interpretation by Translation

A number of (meaning preserving) translations from DRSs to first-order logic formula syntax are known. The "standard" translation can be found in for instance [KR93]. The way these translations work is by translating discourse referents into existential quantifiers, except if they appear in the antecedent DRS of an implicational condition—in that case they are translated into universal quantifiers. DRS-conditions are translated into a conjunction of their respective translations.

The standard translation covers all constructs of our core DRS language, except the modal DRS-conditions □B, ◇B and x:B. To deal with these, we need to use a technique called reiteration, a well-known method to translate modal logics into first-order logic. The translations of these modal expressions are defined, with respect to a translation function $(\mathrm{x},\phi)^t$, in the following way:

$$
\begin{aligned}
(\mathrm{v},\Box\mathrm{B})^t &= \forall \mathrm{w}(\mathrm{R}(\mathrm{v},\mathrm{w})\rightarrow(\mathrm{w},\mathrm{B})^t)\\
(\mathrm{v},\Diamond\mathrm{B})^t &= \exists \mathrm{w}(\mathrm{R}(\mathrm{v},\mathrm{w})\wedge(\mathrm{w},\mathrm{B})^t)\\
(\mathrm{v},\mathrm{x}{:}\mathrm{B})^t &= (\mathrm{R}(\mathrm{v},\mathrm{x})\wedge(\mathrm{x},\mathrm{B})^t)
\end{aligned}
$$

For the full translation the reader is referred to [Bos04]. From a computational point of view, the standard translation is preferred to the "modal" translation, for the latter introduces additional quantifiers and relations in the output formula. This simple insight is exploited in the implementation, where the standard translation is used unless the input DRS contains one of the modal DRS conditions.

2.3 Extending the Core DRS Language

Several extensions to the core DRS language are required to deal with semantic underspecification and to combine it with the lambda calculus (see next section) for semantic construction. We will use the binary α operator to indicate presupposed and anaphoric information, and the binary ; operator to indicate merge of DRSs. Both operators take DRSs as arguments. Ambiguity resolution will always result in a DRS built over the core DRT ingredients (i.e. those shown in clauses 1–6 above), i.e. a DRS that can be translated to first-order logic.

We will adopt Van der Sandt's "presupposition as anaphora" theory [VdS92] to analyse presupposition triggers, including determiners (such as the, both, another, all), possessives, pronouns, presuppositional adjectives (such as other, previous, new), and proper names. Elementary presuppositions are marked in the lexicon by the binary α-operator, indicating that its first argument is a presupposed DRS, and its second argument an asserted DRS. Consider for instance the DRS for 'it includes another country' (as usual we will show DRSs in the box notation for convenience):

$$(\begin{array}{|c|}\hline x\\\hline \textsc{neuter}(x)\\\hline\end{array}\ \alpha(\begin{array}{|c|}\hline y\\\hline \textsc{country}(y)\\\hline\end{array}\ \alpha\ \begin{array}{|c|}\hline z\ e\\\hline \textsc{country}(z)\ \ z\neq y\\ \textsc{include}(e)\\ \textsc{agent}(e,x)\ \textsc{theme}(e,z)\\\hline\end{array}))$$

DRSs constructed with the α operator are subject to presupposition resolution, and can therefore be seen as underspecified semantic representations. Following Van der Sandt, presuppositions can either be resolved by binding them to a suitable antecedent or by accommodating them on an accessible level of DRS. We use the algorithm described in [Bos03] to implement presupposition resolution. Resolved DRSs are defined over the core DRS syntax, and hence can be translated into first-order logic, which in turn makes it possible to implement logical reasoning (see Section 4).

3 The Syntax-Semantics Interface

3.1 Combinatory Categorial Grammar and DRT

Combinatory Categorial Grammar (CCG) is a type-driven lexicalised theory of grammar based on categorial grammar [Ste01]. The lexical entries in CCG pair a syntactic category (defining syntactic valency and directionality) with a semantic representation. We will use the λ-calculus as a tool for building semantic representations compositionally, and combine it with the DRS language introduced before. Consider some example lexical entries (@ denotes functional application):

$$\text{a: } \langle \texttt{NP/N},\ \lambda p\lambda q.(\begin{array}{|c|}\hline x\\\hline \ \\\hline\end{array};p@x;q@x)\rangle \qquad \text{man: } \langle \texttt{N},\ \lambda x.\begin{array}{|c|}\hline \ \\\hline \textsc{man}(x)\\\hline\end{array}\rangle$$

$$\textsf{lied: } \langle \texttt{S\textbackslash NP}, \lambda n.(n@\lambda x. \begin{array}{|c|}\hline \text{e} \\ \hline \textsc{lie}(e) \quad \textsc{agent}(e,x) \\ \hline \end{array})\rangle$$

Combinatory rules project such lexical categories onto derived categories. The CCG formalism we use employs forward and backward application, generalised forward composition, backward composition, generalised backward-crossed composition, and type raising. There is also a coordination rule conjoining categories of the same type. This inventory of rules allow free derivation of non-standard constituents, supporting various cases of relativization and coordination [Ste01]. Consider a simple derivation using the lexical entries given above (applying β-convertion and merge reduction when possible):

$$\textsf{a man: } \langle \texttt{NP}, \lambda q.(\begin{array}{|c|}\hline \text{x} \\ \hline \textsc{man}(x) \\ \hline \end{array};q@x)\rangle \qquad \textit{(forward app.)}$$

$$\textsf{a man lied: } \langle \texttt{S}, \begin{array}{|c|}\hline \text{x e} \\ \hline \textsc{man}(x) \quad \textsc{lie}(e) \quad \textsc{agent}(e,x) \\ \hline \end{array}\rangle \qquad \textit{(backward app.)}$$

3.2 The Clark and Curran Parser

Previous work has shown that standard statistical techniques can be used for practical wide-coverage parsing with state-of-the-art performance, despite the derivational ambiguity allowed by CCG. A number of parsers have been developed, which all use a grammar derived from CCGbank [HS02], a treebank of normalform CCG derivations derived semi-automatically from the Penn Treebank. In the work described in this paper we used the Clark and Curran parser [CC04] to generate DRSs.

The Clark and Curran parser takes a POS tagged sentence as input and assigns categories using a CCG supertagger, leading to a highly efficient and robust parser. The lexical categories used by the parser consists of those occurring at least ten times in the CCGbank, resulting in 409 categories. For about 300 (the most frequent) of these categories we assigned a semantic representation in the form of a λ-DRS. (In principle, nothing stops us to assign semantic representations to all 409 categories.)

This relatively small set can be used to create a robust and accurate DRS parser. Higher coverage can be achieved by a semantic template for parts of the derivation tree that could not be analysed (due to unsupported categories). This was done for the ten most frequent categories. This has the welcoming effect that even if some of the categories are unknown, there will still be a semantic analysis in many cases for the remaining part of the sentence.

Apart from the derivation rules mentioned before, the parser uses also a small set of type-changing rules and a couple of punctuation rules taken from CCGbank. Hence, for a given input sentence, the automatically extracted grammar can produce a very large number of derivations. A packed chart is used to represent the derivation space, and efficient algorithms are used for finding the most probable derivation [CC04].

3.3 Semantic Construction

The output of the Clark and Curran parser is a normal form derivation coded as a tree structure, where the leaves represent lexical items and the nodes represent one of the CCG combinatorial rules, a unary type-changing rule, or a punctuation rule. Given that we have assiged appropriate semantic representations to the lexical items, translating the normal form derivation into a DRS is a process consisting of the following tasks:

1. interpreting the combinatorial rules in terms of functional application;
2. dealing with the type-changing rules and punctuation rules;
3. applying β-reduction to the resulting tree structure (producing a DRS).

The first task deals with the CCG combinatorial rules. The rules we currently use are forward application (FAPP), backward application (BAPP) , generalised forward composition (GFCOMP), backward composition (BCOMP), generalised backward cross composition (GBCROSS), and type-raising (TR). These are interpreted in terms of the lambda calculus along the following rules:

$$\begin{array}{rcl}
\textsc{fapp}(\phi,\psi) & = & (\phi@\psi) \\
\textsc{bapp}(\phi,\psi) & = & (\psi@\phi) \\
\textsc{gfcomp}(\phi,\psi) & = & \lambda\vec{x}.(\phi@(\vec{x}@\psi)) \\
\textsc{bcomp}(\phi,\psi) & = & \lambda x.(\psi@(x@\phi)) \\
\textsc{gbcross}(\phi,\psi) & = & \lambda\vec{x}.(\psi@(\phi@\vec{x})) \\
\textsc{tr}(\phi) & = & \lambda x.(x@\phi)
\end{array}$$

The second task deals with the type-changing and punctuation rules. Currently no semantic interpretation is given to the punctuation rules. The type-changing rules are dealt with by looking up the specific rule and replacing it by the resulting semantics. For instance, the rule for bare nouns that changes category `N` to `NP` converts the semantics ϕ as follows:

$$\textsc{typechange}(\texttt{N},\texttt{NP},\phi) = \lambda\text{p}.(\begin{array}{|c|}\hline \text{x} \\ \hline \\ \hline \end{array};\phi@\text{x};\text{p}@\text{x})$$

Tasks 1–2 are implemented using a recursive algorithm that traverses the normal form derivation as output by the parser and returns a λ-expression. Task 3 reduces this to the target semantic representation by applying β-reduction to this λ-expression. The system is able to output underspecified DRSs as well as fully resolved DRSs, either in Prolog format or in XML. Some example output of our system is shown in Figure 1.

```
sem(7,

 [word(7001,'Mubarak'),word(7002,reviewed),word(7003,the),word(7004,blueprints),word(7005,for),word(7006,a),
  word(7007,number),word(7008,of),word(7009,other),word(7010,huge),word(7011,national),word(7012,projects),
  word(7013,','),word(7014,known),word(7015,as),word(7016,'Egypts'),word(7017,'21st'),word(7018,century),
  word(7019,project),word(7020,'.')],

 [pos(7001,'NNP'),pos(7002,'VBN'),pos(7003,'DT'),pos(7004,'NNS'),pos(7005,'IN'),pos(7006,'DT'),
  pos(7007,'NN'),pos(7008,'IN'),pos(7009,'JJ'),pos(7010,'JJ'),pos(7011,'JJ'),pos(7012,'NNS'),pos(7013,','),
  pos(7014,'VBN'),pos(7015,'IN'),pos(7016,'NNS'),pos(7017,'JJ'),pos(7018,'NN'),pos(7019,'NN'),pos(7021,'.')],

 alfa(nam,drs([7001:A],[7001:pred('Mubarak',[A]),7001:ne(A,'I-PER')]),
          alfa(def,drs([7003:B],[7004:pred(blueprint,[B])]),
                   merge(drs([7006:C],[7007:pred(number,[C])]),
                         merge(merge(drs([7009:D],[]),
                                     alfa(def,drs([0:E],[7010:pred(huge,[E]),7011:pred(national,[E]),
                                                         7012:pred(project,[E])]),
                                              drs([],[7009:not(drs([],[0:eq(D,E)])),
                                                      7010:pred(huge,[D]),7011:pred(national,[D]),
                                                      7012:pred(project,[D])]))),
                               drs([7014:F,7016:G,7002:H],[7008:pred(of,[C,D]),7014:pred(know,[F]),
                                                           7014:pred(patient,[F,C]),7016:pred(egypt,[G]),
                                                           7017:pred('21st',[G]),7017:ne(G,'I-DAT'),
                                                           7018:pred(century,[G]),7018:ne(G,'I-DAT'),
                                                           7019:pred(project,[G]),7015:pred(as,[F,G]),
                                                           7005:pred(for,[B,C]),7002:pred(review,[H]),
                                                           7002:pred(agent,[H,A]),7002:pred(patient,[H,B])])))))).
/*
  ______________    _______________    ____________     ______________   ________________    ________________
 | x1           |  | x2            |  | x3         |   | x5           | | x4             |  | x7 x8 x6       |
 |--------------|  |---------------|  |------------|   |--------------| |----------------|  |----------------|
(| Mubarak(x1)  |A(| blueprint(x2) |A(| number(x3) |;((| huge(x5)     |A|      _________ |);| of(x3,x4)      |)))
 | ne(x1)=I-PER |  |_______________|  |____________|   | national(x5) | |     |         || | know(x7)       |
 |______________|                                      | project(x5)  | | __  |---------|| | patient(x7,x3) |
                                                       |______________| |   | | x4 = x5 || | egypt(x8)      |
                                                                        |     |_________|| | 21st(x8)       |
                                                                        | huge(x4)       | | ne(x8)=I-DAT   |
                                                                        | national(x4)   | | century(x8)    |
                                                                        | project(x4)    | | as(x7,x8)      |
                                                                        |________________| | project(x8)    |
                                                                                           | for(x2,x3)     |
                                                                                           | review(x6)     |
                                                                                           | agent(x6,x1)   |
                                                                                           | patient(x6,x2) |
                                                                                           |________________| */
```

Figure 1: Example DRS output in Prolog format and pretty print for Mubarak reviewed the blueprints for a number of other huge national projects, known as Egypts 21st century project.

4 Semantic Interpretation

4.1 Reasoning

Computational semantics is only computational semantics if one is able to perform reasoning with the representations that are produced. Reasoning can come in various ways for different NLP applications. We will here concentrate on one test: consistency checking [BBKdN01]. A DRS is said to be consistent if it is satisfiable in a model. A DRS is inconsistent if its negation is valid, i.e. satisfiable in any model. Using the translation to first-order logic, we can test this using a model builder and a theorem prover. We translate the DRS into a first-order formula ϕ, and if a model builder is able to construct a model for ϕ we know that the DRS is consistent. If a theorem prover is able to show that $\neg\phi$ is a theorem, we know that the DRS is inconsistent.

Now, inferences without any appeal to background knowledge (lexical knowledge, ontological knowledge, world knowledge, situational knowledge) are rather dull and not very useful. We need to incorporate some background knowledge relevant to the DRS and use that to support the consistency checking task. For small, closed domains it is relatively easy to pin down background knowledge. For a wide-coverage system, we need a robust method to approximate this. We will use the relations in WordNet [Fel98] to achieve this. Of course, to compile the entire WordNet into first-order logic and give that to a theorem prover won't bring us anywhere—theorem provers are designed to deal with mathematical problems, they are not good at dealing with huge chunks of formulas. We need to chop down WordNet to a database that just covers the concepts appearing in the DRS. We call these subsets of WordNet *MiniWordNets* and the next section shows how we construct them.

4.2 Calculating Background Knowledge

Calculating background knowledge is a difficult problem of which we only will scratch the surface. Even though the availability of a rich resource such as WordNet gives us a firm push in the right direction, there are a number of issues we won't consider in the scope of this paper. Although we won't deal with word-sense disambiguation or words that are not found in WordNet, we show that we actually can automatically generate background knowledge in the form of small ontologies.

The algorithm for generating a MiniWordNet takes as input a DRS and outputs world knowledge in the form of first-order axioms. It works as follows. We take all symbols used in the DRS that corresponds to common nouns or proper names. This can be done easily because the extended DRS format has pointers to the parts of speech of the different words that contributed to its compositional meaning (see Figure 1). For each of the symbols (which correspond to the lemmas of the original words) we generate its hypernyms, using only the most frequent sense of that word. The hypernym relations of the words are then taken to compute an "isa-hierarchy", where the nodes are WordNet synsets and the edges represent isa-relations. The directed graph that we get in this way is further chopped down by removing all redundant concepts, using the following rules:

1. Replace ISA(A,B) and ISA(B,C) by ISA(A,C) if there is no X A≠X such that ISA(X,B). Don't do this if A is a terminal node.

2. Remove ISA(A,B) if there is an X such that ISA(X,A), and there is no X such that ISA(B,X), and there is no X A≠X such that ISA(X,B).

3. Add ISA(X,top) for all X such that there is no Y such that ISA(X,Y).

Rule 1 removes a non-branching node between two other nodes. This is safe to do, as it won't add any inferential power, in fact it will only increase the amount of background knowledge which is bad from the theorem prover's point of view. We don't apply rule 1 to terminal nodes to avoid losing information about synonyms. Rule 2 chops off non-branching top elements of the graph. Again, this is fine as they add nothing to the inferential process. Rule 3, finally, adds a common top concept to turn everything into a connected graph.

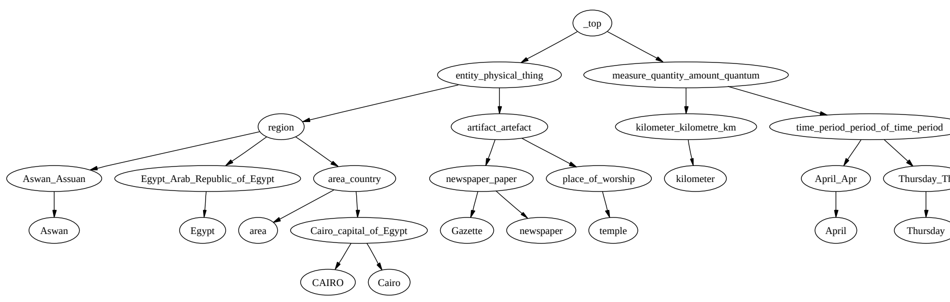

Figure 2: MiniWordNet for CAIRO , April 15 (Xinhua) – Egypt has planned to maintain 12 ancient temples in the area of Aswan, 700 kilometers south of Cairo, the Egyptian Gazette newspaper reported Thursday.

Figure 2 gives an example of such a structure. This structure can be translated into first-order logic and produce three kinds of information. Each $\textsc{isa}(A,B)$ is translated as $\forall x(A(x)\rightarrow B(x))$. Two terminals A and B that stand in the isa-relation to the same concept must be synonyms, hence are translated as $\forall x(A(x)\leftrightarrow B(x))$. Finally, two non-terminal nodes A and B with the same mother nodes are disjoint, yielding $\forall x(A(x)\rightarrow \neg B(x))$. For the isa-hierarchy in Figure 2 we get (among others) the following axioms:

$\forall x(\textsc{Cairo}(x)\rightarrow\textsc{Cairo\_capital\_of\_Egypt}(x))$
$\forall x(\textsc{newspaper\_paper}(x)\rightarrow\textsc{artifact\_artefact}(x))$
$\forall x(\textsc{Cairo\_capital\_of\_Egypt}(x)\rightarrow\textsc{area\_country}(x))$
$\forall x(\textsc{area\_country}(x)\rightarrow\textsc{region}(x))$
$\forall x(\textsc{region}(x)\rightarrow\textsc{entity\_physical\_thing}(x))$
$\forall x(\textsc{artifact\_artefact}(x)\rightarrow \neg\textsc{entity\_physical\_thing}(x))$

4.3 Inference

Since we use first-order logic, there is no need to build ourselves an inference engine. There are many theorem provers and model builders available for first-order logic, and for the work described in this paper, we chose to work with the model builder Paradox 1.0 [CS03] and the theorem prover Vampire 7.0 [RV02]. (The motivation to pick these inference engines is bluntly because they are currently the best inference tools for first-order logic, at least according to the results of the latest CADE ATP system competition, `http://www.cs.miami.edu/~tptp/CASC/`.) We use Paradox to check whether a DRS is consistent, and Vampire to check whether the DRS is inconsistent. Ideally these tests are carried out in parallel, in order to be able to halt the theorem prover when a counter-model is found and vice versa. We ran a test on one of Section 00 of the WSJ corpus, to verify the consistency of the semantic representations and the calculated background knowledge. We were able to produce DRSs and MiniWordNets for 96.0% of the (1902) sentences. In 88% of the cases Paradox found a model, meaning that the DRS and background knowledge was consistent. In 12% cases Vampire found a proof, indicating contradictory information. (In none of the cases neither the theorem prover nor the model builder came back with a result.) It is interesting to look at the inconsistent cases. They were caused by errors in anaphora and presupposition resolution, incorrect CCG derivations, and inadequate semantic representations. These findings give us good indications where to improve the system. Put differently, we have a system that automatically detects its own mistakes using off-the-shelf inference.

5 Conclusion and Future Work

We showed that it is possible to have a robust and wide-coverage system that generates semantic representations with relevant background knowledge from texts and perform first-order inferences on the result. The key feature of this approach is its wide coverage. We don't know any work that converts raw text into logical form and performs inferences with the results, achieving a coverage of 96% (of which 88% logically consistent).

We would like to be in a position to be able to further evaluate the approach, in particular to test semantic accuracy and semantic adequacy (it is important to distinguish these two notions). For now, it is impossible to measure semantic accuracy as there is no corpus with gold standard representations that would make comparison possible. Ideally a gold standard would be independent from any particular semantic formalism but whether it is possible to arrive at such an annotation scheme remains far from unclear, as there is little consensus on what constitutes the ideal semantic formalism in computational semantics.

Measuring semantic adequacy can in principle be done by running the system on controlled inference tasks for selected semantic phenomena. We know two of such test suites. The first is one developed in the FRACAS project [CCVE+96]. It is however inspired by examples taken from textbooks on formal semantics, covering a lot of complex phenomena related to generalized quantifiers, plurals, comparatives, attitudes, anaphora, ellipsis, and temporal phenomena. We are just not ready to perform satisfactorily on such a test suite with a system that emphasises high coverage rather than semantic precision. The second is the "Recognising Textual Entailment Challenge" evaluation exercise that just has been proposed as part of the PASCAL Challenges Program. We envisage to say more about the semantic adequacy of our system by participating in this task.

Although we are aware that this is just the beginning of an architecture for robust wide-coverage semantic interpretation, and there is a lot of work to be done to get a system with high semantic precision, we also believe that the system presented in this paper is already of great value for numerous NLP applications. Existing applications such as question answering, summarisation and machine translation are likely to be improved by using more detailed semantic representations, which we aim to achieve in future work.

References

[BBKdN01] Patrick Blackburn, Johan Bos, Michael Kohlhase, and Hans de Nivelle. Inference and Computational Semantics. In Harry Bunt, Reinhard Muskens, and Elias Thijsse, editors, *Computing Meaning Vol.2*, pages 11–28. Kluwer, 2001.

[BCS+04] J. Bos, S. Clark, M. Steedman, J.R. Curran, and Hockenmaier J. Wide-Coverage Semantic Representations from a CCG Parser. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, Geneva, Switzerland, 2004.

[Bos03] J. Bos. Implementing the Binding and Accommodation Theory for Anaphora Resolution and Presupposition Projection. *Computational Linguistics*, 2003.

[Bos04] J. Bos. Computational Semantics in Discourse: Underspecification, Resolution, and Inference. *Journal of Logic, Language and Information*, 12(2), 2004.

[CC04] S. Clark and J.R. Curran. 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, 2004.

[CCVE+96] R. Cooper, R. Crouch, J. Van Eijck, C. Fox, J. Van Genabith, J. Jaspars, H. Kamp, M. Pinkal, D. Milward, M. Poesio, and S. Pulman. Using the Framework. Technical report, FraCaS: A Framework for Computational Semantics, 1996. FraCaS deliverable D16.

[CS03] K. Claessen and N. Sörensson. New techniques that improve mace-style model finding. In *Model Computation – Principles, Algorithms, Applications (Cade-19 Workshop)*, Miami, Florida, USA, 2003.

[Fel98] C. Fellbaum, editor. *WordNet. An Electronic Lexical Database.* The MIT Press, 1998.

[HS02] J. Hockenmaier and M. Steedman. Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, PA, 2002.

[KR93] H. Kamp and U. Reyle. *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT.* Kluwer, Dordrecht, 1993.

[RV02] A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*, 15(2–3), 2002.

[Ste01] M. Steedman. *The Syntactic Process.* The MIT Press, 2001.

[VdS92] R.A. Van der Sandt. Presupposition Projection as Anaphora Resolution. *Journal of Semantics*, 9:333–377, 1992.