Classifying Scope Ambiguities

Malte Gabsdil and **Kristina Striegnitz**
Department of Computational Linguistics
Universität des Saarlandes, Saarbrücken, Germany
{gabsdil,kris}@coli.uni-sb.de

Abstract

We describe the architecture and implementation of a system which compares semantic representations of natural language input w.r.t. equivalence of logical content and context change potential. Giving a clear graphical representation of the relationship between different readings, the stand-alone version of the system can be used as a classroom tool. Furthermore the core system can be incorporated into other discourse processing systems (e.g. Johan Bos' DORIS system (Bos, 1998)) where one might want to ignore logically equivalent readings in order to keep the number of readings small and thus improve efficiency.

The system relies heavily on existing implementations and code available via the internet. These are integrated and put to the desired use by a Prolog interface. By illustrating the architecture of this system, we want to argue that it is possible to build rather complex systems involving multiple levels of linguistic processing without having to spend an unreasonably large amount of time on the implementation of basic functionalities.

1 Introduction

Scopal ambiguity arising from the interaction of several quantifiers in one sentence has been a major topic of investigation (Cooper, 1975; Hobbs and Shieber, 1987) in formal semantics. But although it has been shown that all possible permutations of the quantifiers need not give rise to well-formed readings, in principle the number of readings is exponential in the number of quantifiers in a sentence. Hence, enumerating all readings can be quite inefficient. Lately, underspecification has been investigated intensively as a possible solution to this problem. The idea is to find a representation which describes all readings in a compact way and to delay explicitly spelling out these readings for as long as possible. However, sometimes enumeration cannot be avoided; for example when inference is to be done. Direct deduction, i.e. deduction on underspecified descriptions, is being investigated, but there are still a lot of unsolved problems. In contrast, normal deduction on e.g. formulas of first order logic is well understood. There is a long tradition of research on automated theorem proving in computer science and many efficient implementations are available.

The problem of exponential growth of the number of readings and thus the number of inference problems, can be mitigated by collapsing logically equivalent readings into one group. Everything that can be proved to be true for one arbitrary member of such a group should also hold for all others (though, as we shall see below, members of a group can differ in their *dynamic* potential).

We describe a system which takes natural language discourse as input, enumerates all possible readings, and orders them in a graph like structure. This graph gives a clear representation of equivalence of readings and the entailment relationships that hold between different readings. Furthermore the dynamic potential of each reading is computed. It is also possible to compare the semantic content of two different input sentences, again in terms of their static and dynamic meanings.

The linguistic modules, grammar, semantic construction, and enumeration of readings, are based on code presented in (Blackburn and Bos, 1998), which is available via the World Wide Web. The comparison of readings as to their logical equivalence is formulated as entailment problems which are then send to theorem provers. For this task we have a local installation of Otter (McCune, 1994), but can also make use of the MathWeb society of theorem proving agents (Franke and Kohlhase, 1999).

By using existing systems and assembling them in a "plug and play" fashion, we were able to build a system which makes use of state of the art techniques and covers several levels of linguistic processing. And we could do so without having to devote unreasonably much time to the development of the basic modules, before being able to turn to the subjects we were originally interested in.

The result is a tool which, due to its modularity, is well suited for further extension. Integration with other systems (e.g. the DORIS system (Bos, 1998)) is planned. Also, we think that it should be a nice classroom tool to illustrate the effect of scopal ambiguity and demonstrate how logically equivalent sentences may differ in their dynamic potential.

2 Architecture

The System consists of three different components that are linked together by a Prolog interface (see Figure 1).

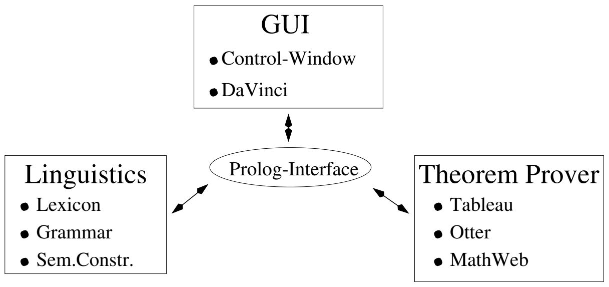


Figure 1: Schematic view of the System-Architecture

The linguistics module is based on code presented in (Blackburn and Bos, 1998). It comprises a 150 word lexicon and a grammar for a small fragment of English. As for semantic construction, there are several different formalisms included: Keller-Storage (Keller, 1988) and Hole Semantics (Bos, 1995) applied to predicate logic as well as DRT. All these formalisms come with their own interface to the lexicon and can therefore use the same grammar.

The second module consists of interfaces to different theorem provers. First, a simple tableau-prover again taken from Blackburn and Bos (1998). Second, a local installation of Otter (McCune, 1994), a resolution based theorem prover. Finally, we can make use of the MathWeb society of distributed theorem proving agents (Franke and Kohlhase, 1999), which gives us access to a variety fast theorem provers.

The third module of the system is the user interface, which again is subdivided into two parts. First, a control panel which accepts user input and offers menus for selecting different semantic formalisms and theorem provers. Second an interface to the DaVinci system developed at the University of Bremen (Fröhlich and Werner, 1998). DaVinci is a tool for representing trees and lattice-structures. It is well suited for our purposes because we represent implication relations between readings in terms of semilattices (see section 4). By clicking on the nodes of such a lattice, one can see the readings and (according to the chosen semantic formalism) DRSs that are associated with them.

3 The Prolog interface

The Prolog interface has to read the user input and interpret the chosen options. There are two main tasks between which the user can choose: firstly, analyse one discourse and order the different readings it may have according to logical implication, and secondly, take two discourses as input and compare them w.r.t. logical implication. Furthermore the user can specify a semantic formalism and a theorem prover.

As a first step, independent of the chosen task, the input is analysed by the linguistics module. The analyses are returned in form of lists of readings, which may be logical formulae or DRSs, depending on which semantic formalism was specified. In the second step the readings are sorted. The core sorting algorithm is the same for both tasks. It takes a list of first-order logic formulae as input and returns a graph-like structure expressing the entailment relationships that hold between these readings. Depending on the chosen semantic formalism in a third step the dynamic potential of each reading is computed.

In the following we describe the sorting of readings and the computation of their dynamic potential in more detail.

3.1 Sorting of readings

The entailment relationship between two readings, r1 and r2, can be determined by sending the problems "r1 implies r2" and "r2 implies r1" to a theorem prover. The answers will tell us whether they are equivalent, one implies the other, or neither is the case.

As mentioned above, the input to the sorting algorithm is a *list* of readings. If we were to compare each reading with all others, we would have to make $2\sum_{x=1}^{n-1} x$ proofs, where n is the number of readings. The worst case scenario will always be this, but by choosing a clever representation, it is possible to often get along with less proofs.

We chose a graph-like structure[1]. Readings are associated with nodes of this graph and entailment is encoded via the dominance relation between nodes, which works like this: Let r1 and r2 be two readings which are associated with nodes n1 and n2 respectively. r1 implies r2 iff n1 dominates n2.

Each reading is given a unique identifier and then nodes are represented by the following structure.

$$\begin{bmatrix} \texttt{label} & \textit{node id} \\ \texttt{ids} & \textit{set of reading ids} \\ \texttt{up} & \textit{set of node ids} \\ \texttt{down} & \textit{set of node ids} \\ \texttt{other} & \textit{set of node ids} \end{bmatrix}$$

The feature `ids` contains the ids of all readings which are associated with this node. Naturally this means that they are all equivalent. `up` indicates those readings that imply the readings associated with this node and `down` those that are implied by this node. `other` holds the rest, i.e. those readings that are not in an entailment relation with the readings of this node.

The graph is described by a list of these node-structures. In the beginning of the sorting process it is empty. Then every reading is compared to one representative of each node in this list (hereby avoiding unnecessary proofs) and the newly gained information is added by extending the appropriate slot of the node structure. If the reading couldn't be inserted into the `ids` slot of any node, a new node is created for this reading and concatenated to the list. Further proofs could be saved by exploiting transitivity during the sorting process. Imagine reading r1 implies reading r2, and we already know that reading r2 implies reading r3, then there is no need to compare readings r1 and r3.

Section 4 contains a detailed example illustrating the structure of our representation.

[1]This is inspired by an idea of Denys Duchier (Duchier and Gardent, 1999)

3.2 Computation of dynamic potential

We represent the dynamic potential of a semantic description as the set of discourse referents that are accessible for a continuation of the discourse, i.e. the discourse referents introduced in the top level DRS. A discourse referent is identified by the things we know about him, so that we just have to collect all the conditions which mention this discourse referent.

It is worth emphasizing that readings may differ in their dynamic potential, although they are logically equivalent. Again, we refer to the following section for an example.

4 Examples and Evaluation

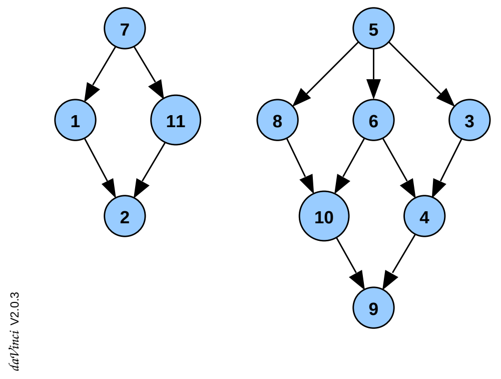
We now want to illustrate by means of an example how collapsing logically equivalent readings into one group reduces the number of proofs necessary to compute an entailment lattice. The example given in (1) has 18 different readings which can be reduced to 11 groups.

(1) Every owner of a hash bar gives every criminal a big kahuna burger.

The graph representing the entailment relationships between these groups has a rather interesting form as it falls into two parts. It is displayed in Figure 2. In Figure 2 we also give the internal representation of nodes `5` and `10`. Note how this representation identifies their position in the graph.

In the worst case, a pairwise comparison of 18 different readings would involve $2\sum_{n=1}^{17} n = 306$ proofs. By grouping equivalent readings together, the system can reduce this number by almost half to 162.

The graph in Figure 2 is split in two, which shows that the readings represented in the left lattice stand in no entailment relation to the readings on the right side. In this particular example, the lattice on the left side holds the readings where *a hash bar* appears in the restriction of *every owner*, meaning that everyone has its own hash bar. In contrast, the lattice on the right comprises all readings where there are several owner of one and the same hashbar, i.e. *a hash bar* outscopes *every owner*.



$$\begin{bmatrix} \texttt{label} & 5 \\ \texttt{up} & [] \\ \texttt{down} & [8,6,3, \\ & \;10,4,9] \\ \texttt{other} & [7,1,11,2] \end{bmatrix} \quad \begin{bmatrix} \texttt{label} & 10 \\ \texttt{up} & [5,8,6] \\ \texttt{down} & [9] \\ \texttt{other} & [3,4,7, \\ & \;1,11,2] \end{bmatrix}$$

Figure 2: Lattice structure for (1) and internal representation of nodes `5` and `10`

Another interesting example is:

(2) Butch didn't forget to shoot a criminal.

Due to the scopal ambiguity between the existential quantifier and the negation, it has (among others) the two readings displayed in Figure 3[2]. These readings are logically equivalent, but differ in terms of their dynamic potential. Assuming that negation restricts accessibility, the second reading can be continued by *His name was Vincent*, while this is not possible for the first one.

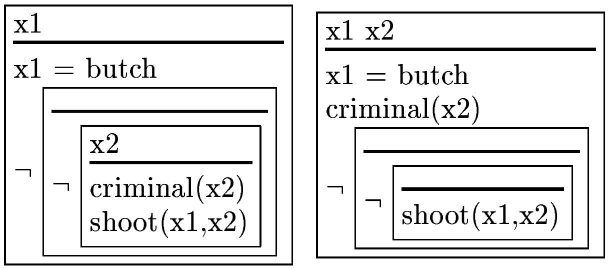


Figure 3: Two readings of Example (2)

[2]We used a somewhat oversimplified analysis of negative verbs, which serves its purpose here, though.

5 Conclusion

We built a system that orders readings of scopally ambiguous sentences w.r.t. logical entailment and dynamic potential. A graph-like representation has proved to be very efficient for this task.

By freely using available implementations we could base our system on advanced techniques in natural language processing and theorem proving.

We are confident that the described system makes up a useful classroom tool. An interesting experiment would be to incorporate it into other discourse processing systems in order to cut down the total number of readings. Also it should be interesting to investigate, how it could be used for selecting the preferred reading of sentences involving presupposition, since it has been argued that always the strongest reading is preferred.

References

P. Blackburn and J. Bos. 1998. *Representation and Inference for Natural Language. A First Course in Computational Semantics.* `http://www.coli.uni-sb.de/~bos/comsem/`.

P. Blackburn, J. Bos, M. Kohlhase, and H. de Nivelle. 1999. Inference and Computational Semantics. In Bunt and Thijsse, editors, *IWCS-3*, Tilburg, NL.

J. Bos. 1995. Predicate Logic Unplugged. In *10th Amsterdam Colloquium.*

J. Bos. 1998. The DORIS-System. `http://www.coli.uni-sb.de/~bos/atp/doris.html`.

R. Cooper. 1975. *Montague's semantic theory and transformational syntax.* Ph.D. thesis, University af Massachusetts at Amherst.

D. Duchier and C. Gardent. 1999. A Constraint-Based Treatment of Descriptions. In Bunt and Thijsse, editors, *IWCS-3*, Tilburg, NL.

A. Franke and M. Kohlhase. 1999. System Description: MathWeb, an Agent-Based Communication Layer for Distributed Automated Theorem Proving. In *Cade '99.*

M. Fröhlich and M. Werner. 1998. The daVinci System. `http://www.informatik.uni-bremen.de/~davinci/`.

J. Hobbs and S. Shieber. 1987. An algorithm for generating quantifier scoping. *Computational Linguistics.*

W. Keller. 1988. Nested Cooper Storage: The proper treatment of quantification in ordinary noun phrases. In Reyle and Rohrer, editors, *Natural Language Parsing and Linguistic Theory.* D. Reidel Publishing Company.

W. McCune, 1994. *Otter Reference Manual and Guide.* `http://www-unix.mcs.anl.gov/AR/otter/`.