*Language and Linguistics Compass* 5/6 (2011): 336–366, 10.1111/j.1749-818x.2011.00284.x

# A Survey of Computational Semantics: Representation, Inference and Knowledge in Wide-Coverage Text Understanding

Johan Bos*
*University of Groningen*

Abstract

The aim of computational semantics is to capture the meaning of natural language expressions in representations suitable for performing inferences, in the service of understanding human language in written or spoken form. First-order logic is a good starting point, both from the representation and inference point of view. But even if one makes the choice of first-order logic as representation language, this is not enough: the computational semanticist needs to make further decisions on how to model events, tense, modal contexts, anaphora and plural entities. Semantic representations are usually built on top of a syntactic analysis, using unification, techniques from the lambda-calculus or linear logic, to do the book-keeping of variable naming. Inference has many potential applications in computational semantics. One way to implement inference is using algorithms from automated deduction dedicated to first-order logic, such as theorem proving and model building. Theorem proving can help in finding contradictions or checking for new information. Finite model building can be seen as a complementary inference task to theorem proving, and it often makes sense to use both procedures in parallel. The models produced by model generators for texts not only show that the text is contradiction-free; they also can be used for disambiguation tasks and linking interpretation with the real world. To make interesting inferences, often additional background knowledge is required (not expressed in the analysed text or speech parts). This can be derived (and turned into first-order logic) from raw text, semi-structured databases or large-scale lexical databases such as WordNet. Promising future research directions of computational semantics are investigating alternative representation and inference methods (using weaker variants of first-order logic, reasoning with defaults), and developing evaluation methods measuring the semantic adequacy of systems and formalisms.

## *1. Introduction*

Computational semantics is an interdisciplinary area combining insights from formal semantics, computational linguistics, knowledge representation and automated reasoning. The main goal of computational semantics is to find techniques for automatically constructing semantic representations from expressions of human language (and *vice versa*), in particular representations that can be used to perform inferences in the service of natural language understanding and generation (Blackburn and Bos 2005). In this article, we review computational semantics from a particular angle, focussing on wide-coverage semantic interpretation of texts.

As a matter of fact, computational semantics is viewed here from a logic-oriented perspective, focusing on representational aspects and reasoning with background knowledge. The resulting article is, admittedly, slightly biased towards the use of classical logic in computational semantics. Nevertheless, I have tried to present a neutral view by pointing out weaknesses of this approach as well as proposing alternatives. Parts of Sections 2 and 4 are strongly influenced by and at some points overlap with Blackburn and Bos (2003), but in general, go beyond it by giving a wider and more up-to-date perspective on the matter.

Computational semantics has seen a couple of interesting developments recently. First of all, the coverage and accuracy of implemented systems is now reaching levels of sophistication and robustness that make formal methods potentially useful in real-world applications such as information retrieval, information extraction, spoken dialogue systems and open-domain question answering (Bos 2008a). Secondly, initial steps have been taken into evaluating systems claiming to have semantic competence. A case in point are the benchmarking campaigns for textual inference capabilities (Dagan et al. 2006), or the recently revived machine-reading competitions (Rodrigo et al. 2010). Thirdly, the use of statistical techniques have entered the field, and are often complementary in functionality to purely symbolic approaches. But it remains an open question on how best to combine, say, distributional semantics with compositionality (Bos and Pulman 2011).

These developments demonstrate how computational semantics has matured in the panorama formed by computational linguistics and natural language processing (NLP), in the nearly 40 years since the ground-breaking work on formal semantics of Richard Montague (1973). Besides introducing elementary issues such as meaning representation, inference methods and the role of background knowledge, this article is written with the aforementioned developments in mind, because I believe this is the right time to combine the insights of traditional methods with those currently emerging in the field.

First, we introduce computational semantics by discussing (in Section 2) what kinds of semantic representation are suitable for capturing the meaning of human language. As we will see, there is no simple answer to this question. It depends to a large extent what linguistic phenomena you wish to analyse (which might be dependent on a particular application) and on the level of detail you wish to analyse these phenomena. We shall take first-order logic as a starting point for illustrating how to analyse the meaning of natural language with the help of semantic representations. However, we will also outline the shortcomings of first-order logic and point to various alternatives. After the discussion of suitable semantic representations, we turn (in Section 3) to the issue of constructing them from expressions of natural language. We review various compositional approaches and give an overview of the various types of natural language ambiguities that one has to take into account. Then, in Section 4, we address the question of how can we use the semantic representations of our choice to automate the process of drawing inferences. We introduce techniques from automated reasoning (theorem proving and finite model generation) to implement consistency and informativeness checks. We also briefly present several alternative inference methods: non-monotonic reasoning, abductive reasoning and reasoning with natural logic. Nonetheless, many of the reasoning tasks require additional background knowledge to make any sensible inferences. In Section 5, we discuss how to incorporate and construct this background knowledge from raw text, semi-structured data or large-scale lexical resources. Building semantic interpretation systems without evaluating them doesn't make much sense, and in Section 6, we present and discuss recent developments in evaluating approaches in computational semantics. Finally, in Section 7, we discuss what we think would be the next big problems that need to be solved in computational semantics.

## *2. Representation Issues*

### 2.1. FIRST-ORDER LANGUAGES

Traditionally, formal semantic analyses of human language typically presuppose formalisms offering much expressive power. A case in point is higher-order logic augmented with modalities, as in Montague Grammar (Dowty et al. 1981; Montague 1973). However, in computational semantics some variant of first-order logic is generally preferred as target semantic representation capturing the meaning of a sentence or text.

This choice is sensible for several reasons. First, as we shall discuss in Section 4, inference engines for first-order logic (such as theorem provers and model builders) now offer levels of performance which make them potentially useful for practical reasoning tasks. Second, as we will see in this section, first-order logic is able to deal (at least to a good approximation) with a wide range of linguistic phenomena.

Put differently, first-order logic offers an attractive compromise between the conflicting demands of expressiveness and inferential effectiveness. It also serves as a good starting point for moving towards formalisms equipped with default reasoning mechanisms, which are popular in artificial intelligence, in particular in the area of knowledge representation and reasoning (Brachman and Levesque 2004). Hence, to make this article self-contained, we briefly review the ingredients of first-order logic.

#### 2.1.1. Ingredients of First-Order Logic

Every first-order language has a vocabulary, telling us which symbols are used and how they are used. Suppose we have the following vocabulary:

| Constants | One-place relations | Two-place relations |
|---|---|---|
| BARBAZOO | EGG | FIND |
| LOLITA | SPOONBILL | SEARCH |
| BARBAPAPA | TURTLE | DROP |
| | DUCK | CATCH |
| | LAUGH | |

Such symbols are often called the non-logical symbols of the language. A one-place relation symbol is called a symbol of arity 1, a two-place relation symbol has arity 2, and so on. Sometimes a constant is referred to as a symbol with arity 0.

The remaining ingredients of a first-order language are a collection of *variables* (x, y, z, and so on), the boolean *connectives* ($\wedge$, $\vee$, $\neg$, $\rightarrow$), the *quantifiers* ($\exists$ and $\forall$), and punctuation devices to group together symbols (the round brackets plus the comma). The variables and constants are the *terms* of the language. Given all this terminology, we can define the formulas of the language of first-order logic as follows:

1. If $R$ is a symbol of arity $n$, and $\tau_1,\ldots,\tau_n$ are terms, then $R(\tau_1,\ldots,\tau_n)$ is a formula.
2. If $\tau_1$ and $\tau_2$ are terms, then $\tau_1{=}\tau_2$ is a formula.
3. If $\phi$ and $\psi$ are formulas, then so are $\neg\phi$, $(\phi \wedge \psi)$, $(\phi \vee \psi)$ and $(\phi \rightarrow \psi)$.
4. If $\phi$ is a formula, and x is a variable, then both $\exists x\phi$ and $\forall x\phi$ are formulas.
5. Nothing else is a formula.

This is the classic formula syntax for first-order logic[1] as we shall use it throughout this article. But there are many other ways to define first-order languages, and we will see some examples below.

Consider two examples of English statements and their translations into the first-order language defined in the previous section, aiming to express the meaning conveyed by each statement:

*Barbazoo found an egg.*
$\exists x(x = \textsc{barbazoo} \land \exists y(\textsc{egg}(y) \land \textsc{find}(x,y)))$

*All spoonbills laughed.*
$\forall x(\textsc{spoonbill}(x) \rightarrow \textsc{laugh}(x))$

The first translation can be paraphrased in English as follows: there exists an entity named Barbazoo,[2] and there exists an entity which is an egg, and the first and second entity are in the two-place 'find' relation. Correspondingly, the second translation can be expressed in natural language as 'if something is a spoonbill, then it laughs'.

When paraphrasing these formulas in English, they seem to capture the intended meaning of the sentences. This is an important step towards our goal of representing meaning, but what we are really after is a method for formally interpreting these formulas. Put differently, given a certain context or situation, when can we say when these formulas express true or false statements? The truth definition method proposed by the famous logician Alfred Tarski does exactly this. This approach is known as model-theoretical interpretation (Tarski 1956).

### 2.1.2. Semantic Interpretation

The term 'model' is used for many different purposes in computational linguistics. What we mean by model here is an abstract realisation of a context or situation. In set-theoretic terms, a model $M$ for a first-order language is an ordered pair $\langle D,F \rangle$ consisting of a domain $D$ and an interpretation function $F$. The domain is just a (non-empty) set of entities. In theory, the domain could contain infinitely many entities, but in practice we usually work with finite models. The interpretation function maps non-logical symbols to members of $D$: a constant is mapped to an entity, a one-place relation symbol is mapped to a set of entities, a two-place relation symbol is mapped to a set of pairs of entities, and so on.

A simple example illustrating this machinery is shown in Figure 1. This model describes a situation with seven entities in its domain. One of these entities is a spoonbill ($d_7$), another is named Barbazoo ($d_1$), the others are all eggs. Three eggs were found by Barbazoo in this model. None of the entities in the domain is laughing. It should be clear that this model describes a situation in which the first of the formulas given above is true,

$$\begin{aligned}
M &= \langle D, F \rangle \\
D &= \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\} \\
F(\text{barbazoo}) &= d_1 \\
F(\text{egg}) &= \{d_2, d_3, d_4, d_5, d_6\} \\
F(\text{spoonbill}) &= \{d_7\} \\
F(\text{laugh}) &= \emptyset \\
F(\text{find}) &= \{(d_1,d_2), (d_1,d_3), (d_1,d_4)\}
\end{aligned}$$

Fig 1. A simple example model *M* with a domain of seven entities.

$$
\begin{array}{lll}
M, g \models R(\tau_1, \cdots, \tau_n) & \textit{iff} & (I^g_F(\tau_1), \cdots, I^g_F(\tau_n)) \in F(R), \\
M, g \models \tau_1 = \tau_2 & \textit{iff} & I^g_F(\tau_1) = I^g_F(\tau_2), \\
M, g \models \neg\phi & \textit{iff} & \text{not } M, g \models \phi, \\
M, g \models (\phi \wedge \psi) & \textit{iff} & M, g \models \phi \text{ and } M, g \models \psi, \\
M, g \models (\phi \vee \psi) & \textit{iff} & M, g \models \phi \text{ or } M, g \models \psi, \\
M, g \models (\phi \rightarrow \psi) & \textit{iff} & \text{not } M, g \models \phi \text{ or } M, g \models \psi, \\
M, g \models \exists \mathrm{x}\phi & \textit{iff} & M, g' \models \phi, \text{ for some x-variant } g' \text{ of } g, \\
M, g \models \forall \mathrm{x}\phi & \textit{iff} & M, g' \models \phi, \text{ for all x-variants } g' \text{ of } g.
\end{array}
$$

Fig 2. The satisfaction definition for First-Order Logic.

and the second is false. We say that this model *satisfies* the first formula, and it doesn't satisfy the second.

The definition of satisfaction can be made more precise, and as a matter of fact, is a crucial link between descriptions (first-order formulas) and situations (models). Formally, the satisfaction definition specifies a three-place relation between a model $M = \langle D,F \rangle$, a formula $\phi$, and a variable assignment $g$. The variable assignment functions can be seen as supplying extra contextual information, because it is a function which maps variables to elements of the domain $D$. The satisfaction relation, symbolised by $\models$, is defined in Figure 2 (where *iff* is short for *if and only if* ).

There are two notions in the definition in Figure 2 that we haven't introduced yet: the function $I$, and the idea of variants of assignments. So let's make these more precise. In the first clause, $I^g_F(\tau)$ is $F$(c) if the term $\tau$ is a constant c, and $g$(x) if $\tau$ is a variable x. In the last two clauses, by an x-variant $g'$ of an assignment $g$ we simply mean an assignment $g'$ such that $g'$(y) = $g$(y) for all variables y ≠ x. Intuitively, variant assignments allow us to pick new values for the variable bound by the quantifier (here x).

Now that we have a clear definition for satisfaction, we are able to define some fundamental inferential concepts such as *consistency*, *contradiction*, *informativeness* and *tautology*.

- A set of first-order formulas $\Phi$ is said to be **consistent** if and only if all of them can be satisfied together in some model with respect to the same variable assignment. That is, $\Phi$ is consistent if it describes a 'realisable' situation, in other words, a situation free of contradictions.
- A set of first-order formulas $\Phi$ is said to be **inconsistent** if and only if all of them cannot be satisfied together in a model with respect to the same variable assignment. That is, it is impossible for $\Phi$ to describe a 'realisable' situation.
- A set of first-order formulas $\Phi$ is **informative** if and only if it is *not* satisfied in all models. That is, $\Phi$ is informative if what it describes rules out some situations.
- A set of first-order formulas $\Phi$ is **tautological** if and only if it is satisfied in all models. That is, $\Phi$ describes a statement that is necessarily true. Using linguistic terminology, $\Phi$ is analytic, as it cannot but describe a true situation.

Note that these inferential notions aren't mutually exclusive. For instance, a consistent formula (or set of formulas) can be a tautology or be informative, and any inconsistent formula (set of formulas) is also informative. Figure 3 visualises the relationships between these characterisations of inference notions, and also includes the linguistic terminology (Matthews 1997) for propositions being *contradictory*, *analytic* (a proposition that cannot but be true) or *synthetic* (a proposition whose truth depends on a specific state of affairs).

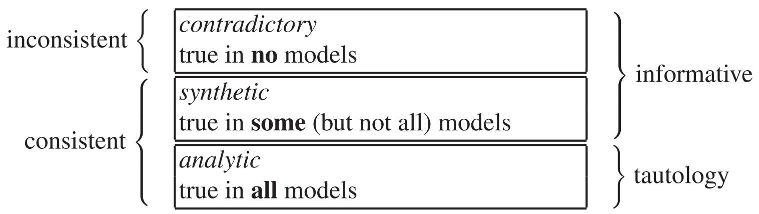

Fig 3. Inferential concepts and linguistic terminology (in italics) explained.

These are theoretical characterisations of inference, and we postpone the discussion of practical inference till Section 4. We must first consider whether first-order logic offers us the kind of expressiveness needed to capture the meaning of natural language expressions. As we shall see, the amount of expressive power and flexibility first-order logic offers opens the way to fine-grained analyses of many semantic phenomena. Nonetheless, we shall also see that there are various limitations that first-order languages face.

### 2.1.3. First-Order Modelling

In the previous section, we have shown some translations from natural language statements to first-order logic, but arguably we haven't touched upon complex linguistic phenomena, such as events, tense, modalities, plurals or anaphora. Is first-order logic capable of describing such semantically rich concepts? The answer is, perhaps surprisingly, to a great extent, in the affirmative. Let's have a look at various possibilities, without pretending to develop a uniform theory of natural language semantics, starting by considering events.

The philosopher Donald Davidson made an influential proposal with respect to events. He argued that many verb phrases in natural language give rise to events. But what *are* events? According to Davidson, despite events being abstract entities, they can be viewed simply as ordinary first-order entities (Dowty 1989). As a consequence, finite verbs introduce existentially quantified variables. Following this idea, we could adapt our vocabulary of non-logical symbols (and, of course, the corresponding models, as they are defined over the vocabulary), and produce the following translation:

*Barbazoo found an egg.*
$\exists x(x = \textsc{barbazoo} \land \exists y(\textsc{egg}(y) \land \exists e\textsc{find}(e,x,y)))$

Note that we adapted our vocabulary: FIND is now a three-place predicate. This simple extension opens a wide number of possibilities. It is now straightforward to deal with sentential and verb phrase modifiers:

*On the beach, Barbazoo found an egg.*
$\exists z(\textsc{beach}\ (z) \land \exists x(x = \textsc{barbazoo} \land \exists y(\textsc{egg}(y) \land \exists e(\textsc{find}(e,x,y) \land \textsc{on}(e,z)))))$

Moreover, this extension also enables us to introduce some simple temporal elements. There are various proposals for dealing with tense and aspect – for a good introduction to modelling tense, see Kamp and Reyle (1993). One way to do this is by introducing first-order entities ranging over time intervals, and adding a two-place relation to the vocabulary linking events to time-intervals, an ordering relation on time-intervals, and a constant NOW denoting the utterance time:

*On the beach, Barbazoo found an egg.*
$\exists z(\textsc{beach}(z) \land \exists x(x{=}\textsc{barbazoo} \land \exists y(\textsc{egg}(y) \land \exists e(\textsc{find}(e,x,y) \land \textsc{on}(e,z) \land \exists t(\textsc{time}(t) \land \textsc{at}(e,t) \land \textsc{before}(t,\textsc{now}))))))$

Again, one might rethink the ontological aspects of modelling events. A popular approach is to take the so-called neo-Davidsonian view on events (Parsons 1990), and introduce thematic roles such as AGENT, PATIENT and THEME. Once more, we can turn this into first-order logic, by modifying the underlying vocabulary:

*On the beach, Barbazoo found an egg.*
$\exists z(\textsc{beach}(z) \land \exists x(x{=}\textsc{barbazoo} \land \exists y(\textsc{egg}(y) \land \exists e(\textsc{find}(e) \land \textsc{agent}(e,x) \land \textsc{theme}(e,y) \land \textsc{on}(e,z) \land \exists t(\textsc{time}(t) \land \textsc{at}(e,t) \land \textsc{before}(t,\textsc{now}))))))$

A different direction of modelling could take modalities (for instance, possibilities and necessities, or going even further, concepts such as believes, desires and intentions) into account. Again we illustrate this with an example without diving too deeply into the matter. Consider the following example (ignoring tense and events for the sake of simplicity):

*Perhaps the egg belongs to a turtle.*
$\exists a(\textsc{world}(a) \land \exists x(\textsc{egg}(a,x) \land \exists w(\textsc{world}(w) \land \textsc{accessible}(a,w) \land \exists y(\textsc{turtle}(w,y) \land \textsc{belongs-to}(w,x,y)))))$

Here we extended our vocabulary with entities called possible worlds, introduced a two-place accessibility relation between possible worlds, and increased the number of arguments of all relations by one. So TURTLE(w,x) could be paraphrased to mean 'x being a turtle in world w'. This idea goes, of course, back to the fundamental notions of modal logic. It is crucial to see that the extensions of modal logic can be modelled by a first-order logic.

Yet another direction to take is to model numeric phrases by extending the vocabulary. This requires a treatment of plural noun phrases, which usually are said to denote sets (Bartsch 1973; Hoeksema 1983; Scha 1984). But one could, for instance, introduce first-order entities standing for groups (as opposed to singletons), and a two-place membership relation $\in$. This would permit translations like the following:

*A spoonbill dropped three eggs.*
$\exists x(\textsc{spoonbill}(x) \land \exists g(\textsc{three}(g) \land$
$\quad \forall y(y{\in}g \rightarrow \textsc{egg}(y)) \land \forall y(y{\in}g \rightarrow \textsc{drop}(x,y))))$

This way of modelling collections would also require further meaning postulates defining the concepts of THREE, and stipulating that spoonbills and eggs are singletons, and that singletons are distinct from groups. This can be done with the following first-order formulas:

$\forall x(\textsc{three}(x) \rightarrow \textsc{group}(x))$ $\qquad$ $\forall x(\textsc{spoonbill}(x) \rightarrow \textsc{singleton}(x))$
$\forall x(\textsc{egg}(x) \rightarrow \textsc{singleton}(x))$ $\qquad$ $\forall x(\textsc{singleton}(x) \rightarrow \neg\textsc{group}(x))$
$\forall g(\textsc{three}(g) \rightarrow \exists x(x{\in}g \land$
$\qquad \exists y(\neg y{=}x \land y{\in}g \land$
$\qquad\quad \exists z(\neg z{=}x \land \neg z{=}y \land z{\in}g))))$

The idea of representing groups as first-order objects goes back to work of Link (1983) and can also be found in description logics (Franconi 1993). One should of course consider the repercussions resulting from extending the ontological structure of the first-order models associated with large groups – associating a sentence such as 'The Irish band U2 made it a night to remember for 60,000 fans' with a first-order model comprising thousands of different entities, one for each U2 fan, is not only impractical, but probably also far from psychological reality. As a matter of fact, efficient reasoning with plural entities is still an open issue in computational semantics (Chaves 2007).

Finally, we draw our attention to modelling anaphoric expressions, and briefly introduce Discourse Representation Theory, DRT (van Eijck and Kamp 1997; Kamp and Reyle 1993). In DRT, the meaning of natural language sentences and short texts are modelled in a Discourse Representation Structure (DRS in short). A DRS can play both the role of semantic *content* and the role of discourse *context*. The content gives us the precise model-theoretic meaning of the text, and the context it sets up serves for establishing anaphoric links between pronouns and their antecedents.

Essentially, a DRS is a first-order representation, but it doesn't have explicit quantifiers to bind variables[3] nor explicit conjunction operators. Instead, a DRS is a structure containing of two parts: a set of *discourse referents* and a set of *conditions*. Let's have a look at a concrete example by considering the following small text spanning two sentences, annotated with a DRS after interpreting the first sentence and the modified DRS after interpretation of the entire text:

*Barbazoo found an egg.*

| x y e . . . |
|---|
| x=BARBAZOO |
| EGG(y) |
| FIND(e) |
| AGENT(e,x) |
| THEME(e,y) |
| ... |

*The spoonbill dropped it.*

| x y e z u e′ . . . |
|---|
| x=BARBAZOO |
| EGG(y) |
| FIND(e) |
| AGENT(e,x) |
| THEME(e,y) |
| SPOONBILL(z) |
| DROP(e′) |
| AGENT(e′,z) |
| THEME(e′,u) |
| u=y |
| ... |

As this example illustrates, a DRS gets updated after each new sentence. Here the pronoun *it* introduces a discourse referent u, which is linked by the equality condition to the discourse referent of the egg, y, introduced in the DRS by the first sentence.

It is important to realise that the core of the DRS language is not more expressive than ordinary first-order logic, and the two formalisms can be freely inter-translated (van Eijck and Kamp 1997; Muskens 1996). In the language of DRSs, logical conjunction and the type of quantification is implicit, and from a practical point of view, this is just a very convenient way to model anaphoric pronouns. Another case in point where DRSs play an essential role is van der Sandt's theory of presupposition projection (van der Sandt 1992). There are, nevertheless, several extensions of the DRS language that go beyond the first-order fragment, and the reader is referred to, for instance, Kamp and Reyle (1993) for generalised quantifiers and Asher (1993) for discourse relations.

### 2.2. LIMITATIONS OF FIRST-ORDER LANGUAGES

We have seen that first-order logic offers expressive power of a degree high enough to capture many aspects of the meaning of human language. We have also seen that it helps to be flexible about the kinds of entities we include in our models. Put differently, first-order approaches go hand-in-hand with rich ontologies. Fine – but are there limitations to this style of first-order modelling? Yes. When we introduce new entities we have to introduce constraints governing how they behave (we called them *meaning postulates* above, and in the next section we shall consider them as axioms, i.e. part of the background knowledge). For example, we might want to constrain the accessibility relation on possible worlds to be reflexive, or constrain the precedence relation on time intervals to be transitive, or insist that groups must have at least two different ordinary individuals as members.

When (as in these examples) the required constraints can be stated in first-order logic, nothing more needs to be said. However if some postulates can't be written in this way, then our first-order modelling is only an approximation. Examples belonging to these class are generic statements: sentences that make a general claim, but are likely to have exceptions (Pelletier and Asher 1997). Statements such as 'cars have four wheels', 'a dog has a tail' and 'birds fly' are called generics. They are usually true, but not always (there are cars with only three wheels, some breeds of dog are tailless and cassowaries are birds unable to fly). First-order logic cannot adequately describe such statements, because it doesn't support any form of non-monotonic reasoning (see Section 4.3.1).

Other examples that go beyond first-order logic are proportional and branching quantifiers. Consider for instance a proportional quantifier such as *most*. Its exact meaning isn't capturable by the mechanisms of first-order logic (just try to describe the meaning of a sentence such as *most eggs hatched* using the quantifiers $\forall$ or $\exists$, and you get the idea), as has been proven by Barwise and Cooper (1981). In addition, Hintikka (1973) claimed that certain English sentences with multiple determiners require branching (rather than linear) quantification. Some of these resulting branching quantifiers can be represented in first-order logic by duplicating some of the logical structure, but others require the expressive power of a second-order logic (Hoeksema 1983; Robaldo 2007; Scha 1984; Sher 1990).

### 2.3. WORKING WITH FRAGMENTS OF FIRST-ORDER LOGIC

After stating that first-order languages lack descriptive power for dealing with certain linguistic phenomena, it comes perhaps as a surprise that several approaches work with impoverished fragments of first-order logic for modelling natural language meaning. Such fragments have less expressive power than fully-fledged first-order logic, but have much better computational properties (they are usually decidable logics, in contrast to first-order logic, which isn't), which explains why they are popular. Pratt-Hartmann (2003) explores theoretical work on fragments of first-order logic for natural language interfaces, and Pratt-Hartmann (2004) defines various fragments of English and their logical complexity.

A further example in this tradition are description logics such as ALE, ALC, SHIQ and OWL, which are now extremely popular in Semantic Web applications (Baader et al. 2002). In description logics, several specialised inference tasks play a role such as instance checking (checking whether a particular entity belongs to a certain concept), relation checking (finding out whether two entities stand in a certain relation), subsumption checking (proving whether a concept is a subset of another concept) or general consistency checking. For an introduction of description logics applied to NLP, see Franconi (2002); for an application to dialogue, see Ludwig et al. (2000); for a natural language interface to an adventure game, see Koller et al. (2004). Most description logics are decidable fragments of first-order logic – some offer facilities such as counting that first-order logic doesn't have built-in.

How high is the price we pay when we trade expressive power for inferential efficiency? As far as we know, very little theoretical work has been carried out to answer this question in detail, with the exception of Pratt-Hartmann's. This is surprising, even more so since description logics such as OWL seem to emerge as standards for the formal tools applied to the semantic web. All we can say is that these restricted formal languages will make a proper analysis of various natural language phenomena, including negation, quantifer scope, and anaphora, a difficult task. The niche for decidable logics in the landscape of computational linguistics seems to be restricted to controlled natural languages.

## *3. Constructing Semantic Representations*

Once we have fixed on a representation formalism (first-order logic, for example, as we do in this article) and a vocabulary, how do we write programs which take human language sentences as input and return semantic representations as output? A common approach is to view this process as one presupposing a syntactic analysis of a sentence whose meaning has to be computed, and assumes that each lexical item contributes to the meaning of the entire sentence. The syntactic analysis, usually producing some kind of tree structure, tells us what kind of lexical items we have, and how they are combined into larger expressions.

One of the key challenges in the syntax–semantics interface is to keep track of variable bindings during the process of meaning composition. This is a technical issue for which various methods have been proposed. Another major challenge is dealing with ambiguities eminent in natural language. This is an issue that manifests itself through various linguistic phenomena, but in order to make it more comprehensible can be divided into lexical, structural and contextual ambiguity. We shall have a look at both of these issues without presenting the problems in detail, but instead provide pointers to good starting points in the literature.

### 3.1. SEMANTIC COMPOSITION AND VARIABLE BINDINGS

The common strategy of producing meaning representations for natural language expressions underlies a so-called *compositional* analysis. This strategy assumes that the meaning of a compound expression is computed from the meanings of its parts. The smallest parts carrying meaning are usually the lexical items (i.e. the words). Given a first-order target representation, providing a meaning representation for a lexical item presents a challenge because for a word in isolation it is hard to specify what the variables of its subcategorised arguments are (if it has any at all). Hence, machinery is needed to perform variable book-keeping operations. This process is further complicated by linguistic phenomena that require renaming of variables in the process of composition, of which coordination and ellipsis are notorious examples. By and large, there are three different approaches tackling this issue: methods based on unification, methods based on the $\lambda$-calculus and methods using linear logic. Let's briefly review them.

The unification-based algorithms are often integrated inside the syntactic formalism to perform the book-keeping of variables, are extremely efficient, but need on the other hand careful consideration to avoid unwanted variable clashes caused by copying or renaming variables (Blackburn and Bos 2005; Moore 1986). A good example of the unification-based approach is found in the minimal recursion semantics (MRS) which is embedded in the Head-driven Phrase Structure Grammar (HPSG) framework (Copestake et al. 2005).

The approaches based on the $\lambda$-calculus and type theory (Church 1940) are less efficient than unification-based methods, but come with a built-in approach of renaming variables whenever necessary, and can also be more easily viewed as independent of the underlying syntactic structure. Montague Grammar is staged within this method (Dowty et al. 1981; Gamut 1991). Computational implementations of the $\lambda$-calculus can be found, for instance, in Blackburn and Bos (2005) and Bos (2008c).

Another interesting approach is to use linear logic as glue language for assembling meaning representations. Linear logic is a 'resource-sensitive' version of classical logic, which means that once you've used a formula to draw a conclusion, you can't use it again. This feature enables it to be used to implement meaning composition in a natural way. Frameworks following this paradigm exist for Lexical Functional Grammar (LFG; van Genabith et al. 1999) and HPSG (Asudeh and Crouch 2002). Assembling meanings with linear logic doesn't presuppose a tree structure of the synactic analysis while methods based on the $\lambda$-calculus do, and is therefore convenient for working with attribute-value structures found in LFG and HPSG.

## 3.2. PROCESSING AMBIGUITIES

Human language is full of vague and ambiguous expressions that logical languages, such as first-order logic, don't have. As a consequence, computing semantic representations for fragments of natural language means dealing with these ambiguities in one way or another. As the focus of this article is on inferential aspects of computational semantics, we won't present any of the technologies proposed for ambiguity processing in full depth. Instead we shall only give a general overview of current directions and provide pointers to relevant work. First of all, we can distinguish between lexical, structural and contextual ambiguity.

With **lexical ambiguity**, we refer to the possible meanings that can be assigned to a simple word – the word senses. Word sense disambiguation (WSD) is an active field within NLP. It is a difficult problem, because the existing standard inventories for word senses employed in WSD tasks, such as WordNet (Fellbaum 1998), are fine-grained, making it hard to develop algorithms that outperform baseline systems that assign the most frequent sense to words. Following Navigli (2009), WSD systems are usually classified as supervised (relying on sense-annotated training corpus) or unsupervised systems (do not require a sense-tagged corpus, instead using resources like dictionaries or thesauri to select senses). Approaches that fall in the first class reach higher accuracy, but are less promising for large-scale use because complete annotation of senses for large corpora do not exist. WSD is an important task for a natural language understanding system, because accurate sense identification is a prerequisite for selecting the right kind of background knowledge.

Other areas of research within computational lexical semantics aim to explain and predict creative usages of words. To these belong metonymy (using an expression to refer to an associated concept) and metaphor (conveying an analogy between two concepts). For example, the standard referent of the term ‘Wall Street’ is a street in New York City; however, in its metonymical use it refers to the American financial and banking industry, which is located at Wall Street. This example is an instance of regular polysemy (Markert and Nissim 2009; Pustejovsky 1995). As a second example, consider the metaphorical use of the noun ‘eye’ in the expression ‘eye of a needle’, which doesn’t refer to an organ of sight (the first sense of ‘eye’ in WordNet), but rather to a small hole (incidentally, the fifth sense of ‘eye’ in WordNet). A recent survey of approaches to metaphors can be found in Shutova (2010).

Lexical semantics is also concerned with the problem of thematic role labelling (see the discussion on neo-Davidsonian event semantics in Section 2.1.3), and thematic roles alone are often seen as a good basis for a shallow semantic representation of texts (Erk and Pado 2006). The existence of large-scale inventories of thematic roles, accompanied with reasonably large annotated texts, of which the most well-known are FrameNet (Baker et al. 1998), VerbNet VerbNet (Kipper et al. 2008) and PropBank (Palmer et al. 2005), has turned thematic role labelling a popular area of research in computational linguistics.

With **structural ambiguity**, we refer to the various ways the recursive structure of a semantic representation can be constructed. A natural language expression that contains more than one scope-bearing element (quantifying noun phrases, modal adverbs, negation particles, etc.) can give rise to a scope ambiguity. Thus, in the example below, there are two possible interpretations: one where a huge flag is hanging in front of all windows, or one where there are as many flags as windows, one hanging in front of each window.

*A flag was hanging in front of every window.*
$\exists x(\textsc{flag}(x) \land \forall y(\textsc{window}(y) \rightarrow \textsc{hanging-in-front-of}(x,y)))$
$\forall y(\textsc{window}(y) \rightarrow \exists x(\textsc{flag}(y) \land \textsc{hanging-in-front-of}(x,y)))$

It has been widely recognised that such scope ambiguities are genuine: firstly, the order in which the scope-bearing operators appear in a sentence doesn’t always reflect the order in the preferred interpretation; and secondly, there isn’t always one general reading subsuming other more specific readings (Blackburn and Bos 2005). Several solutions have been proposed to deal with scope ambiguities at the level of semantic interpretation, of which storage methods (Cooper 1983; Keller 1988) and constraint-based methods of semantic underspecification (Bos 1996; Copestake et al. 2005; Egg et al. 1998; Reyle 1993) are the most popular. Recently, several efforts have been undertaken for devising ways of efficient computing with underspecified semantic representations (Koller and Thater 2010; Koller et al. 2000).

Another instance of structural ambiguity is caused by the collective and distributive interpretation that manifests itself when plural objects are predicated by a verb phrase that can be understood as ranging over groups of individuals. An example of this phenomenon is shown below, where the first translation demonstrates the distributive reading (‘three girls were watching a film each’) and the second shows the collective reading (‘three girls were watching a film together’).

*Three girls were watching a film.*
$\exists g(\textsc{girl}(g) \land \textsc{three}(g) \land \forall x(x{\in}g \rightarrow \exists y(\textsc{film}(y) \land \textsc{watch}(x,y))))$
$\exists g(\textsc{girl}(g) \land \textsc{three}(g) \land \exists y(\textsc{film}(y) \land \textsc{watch}(g,y))))$

The distributive/collective distinction is further complicated by the aforementioned scope ambiguities: in the above examples, there are also readings in which there is a specific film the girls were watching (the wide-scope reading for the noun phrase *a film*), both for the distributive and the collective interpretation. And this isn't the end of the story of structural ambiguities, as certain sentences with several numeric noun phrases give rise to cumulative quantification. Scha (1984) presents the example *600 Dutch firms use 5,000 American computers*, which can be read as the total number of Dutch firms that use an American computer is 600, and the total number of American computers used by a Dutch firm is 5,000. This interpretation requires branching quantification (see Section 2.2).

Finally, with **contextual ambiguity**, we mean those ambiguities that aren't caused by different internal structure nor by lexical choice, but by the context (linguistic or non-linguistic) in which they appear. Anaphoric pronouns are a first example of such ambiguity, as discussed before in Section 2.1.3. The task of anaphora resolution aims to find the correct antecedents for anaphoric expressions (including anaphoric presuppositions, such as definite descriptions; Bos 2003; van der Sandt 1992), for which both the traditional rule-based approaches as well as the more recently advocated statistical approaches offer only approximate solutions (Carter 1987; Mitkov 2002; Soon et al. 2001). It should also be said that hitherto most practical approaches to anaphora resolution focus on singular pronouns (e.g. *he, she, it*), rather than plural pronouns and pronouns referring to abstract entities. Nonetheless, current efforts of creating large-scale annotated resources for anaphora, of which the crowdsourcing method presented in Chamberlain et al. (2008) is a key example, will certainly boost performance of machine learning approaches in the near future.

An additional but related problem is that of coreference resolution, which aims at grouping together noun phrases sharing the same referent, in particular proper names of persons and organisations. The resolution of deixis is concerned with pronouns denoting the speaker and addressee (e.g. *I, we, you*), temporal expressions (e.g. *yesterday, last week*), spatial expressions (e.g. *here*, *there*) and the task of proper name anchoring, for instance that of geographical names (Leidner 2008), can also be viewed as an instance of this class of ambiguities. Finally, elliptical phrases, for instance verb phrase ellipsis in English (Dalrymple et al. 1991; Kehler 1993) cause contextual ambiguities, for which so far only few purely practical approaches exist (Hardt 1997; Nielsen 2005).

## 3.3. WIDE-COVERAGE APPROACHES

The last years have seen the development of wide-coverage, open-domain systems for producing semantic representations for texts. Interestingly, some of them are built on top of various detailed linguistic theories, including:

- HPSG (Copestake and Flickinger 2000);
- LFG (Butt et al. 2002);
- CCG, Combinatory Categorial Grammar (Curran et al. 2007).

There are various implementations based on HPSG. First of all, the LinGO English Resource Grammar (ERG) is a wide-coverage grammar of English based on HPSG (Copestake and Flickinger 2000). The ERG produces underspecified semantic representations, MRS, in which relative scopings of quantifying noun phrases and modifiers remain unresolved (Copestake et al. 2005). Secondly, Sato et al. (2006) translate output of a robust HPSG parser into semantic representations of Typed Dynamic Logic, achieving a reported coverage of approximately 90% for texts of the Penn Treebank corpus. Yet another HPSG-based implementations is the ENJU parser, which comes with a component that produces formulas of predicate logic (Butler et al. 2010).

The system described in Bos (2008c)) produces completely resolved DRSs on the output of a robust parser for CCG (Curran et al. 2007). Crouch and Holloway King (2006) present a rewriting system for a broad-coverage LFG grammar, that produces semantic representations from the functional structures obtained from parsing newswire text. Finally, Moldovan et al. (2007) construct a flat first-order style semantics, 'Logic Forms', on top of a dependency grammar.

These implementations differ substantially from each other, not just with respect to the underlying theory of syntax and parsing approach, but also on the semantic formalism and the manner in which they are constructed. Some of the systems presented below are based on hand-crafted grammars, carefully developed, often over many years (e.g. LFG and HPSG); others are based on robust, statistically driven parsers developed on the basis of the Penn Treebank (Marcus et al. 1993). Some of them produce underspecified semantic representations (i.e. representations in which some ambiguities discussed in the previous section aren't resolved); others fully resolved semantic representations that can be directly interpreted. These differences don't make it easy to compare the various systems.

## 4. *Inference Issues*

### 4.1. INFERENCE IN COMPUTATIONAL SEMANTICS

Inference can play many roles in computational semantics. It can be used in disambiguation to filter out contradictory interpretations or to rank the likelihood of different interpretations. It can be used in generation to test candidate sentences for suitability in a given situation. It can be used in summarisation to check whether a summary contains no new information. It can be used in question answering to find out whether a response is a proper answer to a question. And there are many other applications for which inference is useful.

Many forms of inference are relevant to computational semantics. But, given that we have been advocating first-order logic as a good basis for semantic representation language, we will first discuss inference techniques for first-order logic, by focusing on the use of theorem provers and model builders (Blackburn and Bos 2005). However, first-order logic is known to be undecidable – it is impossible to develop an algorithm that can tell us whether an arbitrary set of first-order formulas is consistent or not, and it is not capable to make any default inferences. It is therefore good to investigate and explore alternatives, and we will discuss default reasoning, abductive reasoning, and reasoning directly on syntactic surface representations.

### 4.2. FIRST-ORDER LOGIC AND INFERENCE

A slight practical concern might be the fact that first-order logic is known to be *undecidable*. What does this mean? This means that it is not possible to design a theorem prover which, when given an arbitrary formula as input, is guaranteed to halt in finitely many steps and correctly classify the input as consistent or not (or for that matter, as informative or not). Despite this, current theorem provers (and to a lesser extent, model builders) are reasonably efficient in practice, reaching levels of performance unheard of a decade ago (Pelletier et al. 2002; Sutcliffe and Suttner 2006; Voronkov 2003). Moreover, inference involving equality was difficult to handle efficiently until recently, but current resolution provers even cope with formulas containing the equality symbol (Riazanov and Voronkov 2002). As semantic representations for human language typically make heavy use of equality (as we have seen in the previous sections), this is an important development for computational semantics.

### 4.2.1. Theorem Proving

Recall that we gave model-theoretic definitions of consistency and informativeness in Section 2. The branch of logic called *proof theory* has developed many ways of reformulating these concepts as symbol manipulation tasks. Proof theory considers the generation of formulas (theorems) from other formulas (axioms) using a set of inference rules; the best-known inference rule is probably modus ponens (from $\{p \rightarrow q, p\}$ derive $q$). Modern automated theorem provers use vastly more sophisticated strategies than this, but in essence they are tools that use symbol manipulation techniques to deal with the tasks of checking for consistency and informativeness.

So how exactly can we use theorem proving for these tasks? Suppose we want to know whether $\varphi$ is consistent. If we give $\neg\varphi$ to a theorem prover, and it finds a proof for this input (i.e. establishes that $\neg\varphi$ is a validity), then we know that $\varphi$ is *not* consistent. (Why? As it proved $\neg\varphi$, this must be true in all models, hence $\varphi$ is false in all models, that is, inconsistent.) On the other hand, if no proof is found (given the resources available), nothing can be said about whether $\varphi$ is consistent or not. If we want to know whether $\varphi$ is consistent with respect to some background knowledge axioms, we can proceed as follows. Let $\Psi$ be the conjunction of a (finite) set of first-order formulas. Then we ask the theorem prover to test the validity of $\neg(\Psi \wedge \varphi)$, or equivalently, $\Psi \rightarrow \neg\varphi$.

Now suppose we want to know whether $\varphi$ is informative. If we give $\varphi$ to a theorem prover, and the theorem prover finds a proof for this input, then we know that $\varphi$ is *not* informative. (Why? since $\varphi$ is a validity, $\varphi$ is true in all models, hence uninformative.) And once again, if no proof is found, all bets are off. Analogously, if we want to know whether $\varphi$ is informative with respect to some background knowledge $\Psi$, we ask the theorem prover to test the validity of $\neg(\Psi \wedge \neg\varphi)$, or equivalently, $\Psi \rightarrow \varphi$. Phrased differently, this is the way to find out whether $\varphi$ is *entailed* by $\Psi$. Summing up: theorem provers offer us a negative handle on the problem of determining consistency, and on the problem of determining informativeness.

Some examples of state-of-the-art theorem provers are Vampire (Riazanov and Voronkov 2002), Spass (Weidenbach et al. 1999), E-KRHyper (Baumgartner et al. 2007) and Otter (McCune and Padmanabhan 1996). Examples of computational semantic applications that include first-order theorem provers are the implementation of the (negative parts of the) consistency and informativeness checks required by Van der Sandt's presupposition resolution algorithm (Blackburn et al. 2001), question interpretation in spoken dialogue systems (Bos and Gabsdil 2000), open-domain question answering (Furbach et al. 2010) and recognising textual entailment (RTE; Akhmatova and Mollá 2006; Bos and Markert 2005). Such practical usages of theorem proving usually have a fall-back strategy in place, particularly in open-domain applications where finding proofs to a large extent depends on the background knowledge available. Furbach et al. (2010), in their application to question answering, use an algorithm that invokes relaxation of the query if no proof of the full query is found. In their system for RTE, Bos and Markert (2005) complement theorem proving with model building as an alternative way of drawing logical inferences. The latter inference method is discussed in the next section.

### 4.2.2. Finite Model Generation

Theorem provers are not the only inference tool available for first-order logic. Another inference technique relevant to computational semantics is finite model generation. (The models that we talk about here are those introduced in Section 2.) Model builders, or model generators, take a first-order formula as input and attempt to build a (finite) satisfying model for it. Thus model builders offer positive handles on both the consistency problem (if one successfully builds a model for $\varphi$, then $\varphi$ must be consistent) and the informativeness problem (if one successfully builds a model for $\neg\varphi$, then $\varphi$ must be informative).

Hence, model building is complementary to theorem proving (which offers, as we have seen, negative handles on both problems), and indeed they are often used in parallel (Blackburn and Bos 2005; Bos and Markert 2005; McCune 1998). This is a good point to briefly return to the situation of undecidability of first-order logic. A theorem prover will say 'no' if an input formula is inconsistent (uninformative), at least, in theory – in practice it might run out of time or memory. A model builder will, however, not always find a model if the input formula is consistent. For this reason, first-order logic is often said to be *semi-decidable*: a theorem prover will prove every theorem, but doesn't always give an answer if the input formula isn't a theorem. Determining whether there is a *finite* model (for a certain model size $n$), however, is decidable. Therefore, to cope with the undecidability problem in practice, it's best to use a model builder in an attempt to find a model up to some pre-specified domain size $n$, while at the same time using a theorem prover to try to determine inconsistency of the input.

Compared to theorem proving, finite model building for first-order logic is a relatively new branch of automated reasoning, and it is fair to say that model builders haven't reached the performance levels demonstrated by theorem provers. Nonetheless, the performance of model builders have improved in the last few years, and two good examples of powerful model builders for first-order logic are Mace (McCune 1998) and Paradox (Claessen and Sörensson 2003). Such 'general purpose' model generators can be useful in linguistic applications, as long as the domains of the models are reasonably small.[4]

As said before, the most obvious use for model builders is to use them in tandem with theorem provers to perform consistency and informativeness checks, but this does not exhaust their potential usefulness. It is also interesting to use model builders to construct models of ongoing discourses (linguists may like to view such models as an intermediate level of representation). Because models are essentially non-recursive, flat structures, they form a level of representation from which it is relatively easy to extract content. Moreover, if a model builder finds a model for a description it will typically find (one of) the smallest models possible; that is, it will find a *minimal model*. This is because the algorithms employed by most model builders work by iteration: they start by attempting to generate a model for size $n$, and if that fails, attempt to do it for $n + 1$, and so on.

Ramsay and Seville (2000) seem to have been one of the first to use model builders for the task of text understanding. The approach has been further explored for natural language disambiguation by Gardent and Konrad (2000), Gardent and Webber (2001), and Cimiano (2003). The most obvious computational semantic applications for model builders is to carry out positive tests for consistency and informativeness for such applications as Van der Sandt's presupposition resolution algorithm (Blackburn et al. 2001), and question answering (Bos and Gabsdil 2000). The CURT system includes a model builder (Blackburn and Bos 2005), as does the spoken dialogue interface to a mobile robot of Bos and Oka (2007) to link semantic interpretations to the situations in the real world. Model building is also part of the Nutcracker system for RTE (Balduccini et al. 2008; Bos and Markert 2005).

### 4.3. ALTERNATIVE INFERENCE METHODS

So far, we have only considered the reasoning methods that come with classical logic. Yet, in many cases, these rigid ways of reasoning (such as modus ponens that we mentioned) isn't always suitable for the purposes of natural language understanding.

#### 4.3.1. Non-Monotonic Reasoning

First-order reasoning, such as consistency and informativeness checking, is not always suitable for the purposes of natural language understanding, especially when it comes to interpreting background knowledge rules. Many statements that one would like to have in its background knowledge are *generic* statements, that is, statements that are usually true, but not always. For instance, if we translate the statement *every dog has a tail* into first-order logic as

$$\forall x(\textsc{dog}(x) \rightarrow \exists y(\textsc{tail}(y) \wedge \textsc{have}(x,y)))$$

then we did a fine job in translating it to logic, but we are missing the point that this is a generalisation, for some breeds of dog are tailless.

Default logics form an extension of first-order logic by introducing defeasible inference rules to overcome the problems introduced by generics (Reiter 1980). In default logics, the generic statement would be interpreted as 'typically, a dog has a tail', and translated in an inference rule roughly of the form 'for every dog, if no contradictions arise if we assume it has a tail, we can conclude it has a tail'.

One of the paradigms in default reasoning applied to knowledge representation and natural language applications is Answer Set Programming (ASP; Gelfond 2008). ASP is an extended form of logic programming that combines the default negation with classical negation to model defaults. Applications of ASP for natural language analysis can be found in Lierler and Görz (2006) and Baral et al. (2008).

#### 4.3.2. Abductive Reasoning

Some of the mechanisms of default logics have also points of contact with *abductive reasoning*. Abductive reasoning implements a kind of 'backward modus ponens' – recall that modus ponens is a rule of *deductive reasoning*, as presented in Section 4.2.

In a nutshell, abduction works as follows. Say we have in our background knowledge the rules that all birds fly, and that all birds are animals. Now assume we see a flying animal: what kind of animal is it likely to be? Well, we can conclude, via abductive reasoning, that it probably is a bird. Thus, abduction is a way of giving explanations. In many cases, however, there could be more than one explanation for an observed fact. If our background knowledge also contains axioms for bats, then abductive reading could also conclude that the flying animal is a bat. Picking out the best explanation is also part of the abductive reasoning process, for which various criteria exist. Moreover, explanations need to be consistent with the observed facts and background knowledge; if we see that the flying animal has feathers, 'bat' isn't a good answer for identifying the animal.

Applications of abductive reasoning in NLP can be found in Hobbs et al. (1990), who argues that many linguistic phenomena require abduction, such as predicting coherence relations and bridging references of definite descriptions. Hobbs et al. (1990) weighs the terms in an abductive proof in order to pick the most likely candidate among all available alternatives for explaining an observation. An open issue is how to compute these weights.

#### 4.3.3. Natural Logic

Natural Logic is a family of inference systems that work directly on the syntactic representation of natural language expressions, without resorting to an intermediate translation to logical formulas. Instead, a Natural Logic system defines an ordering relation on syntactic trees on the basis of their local entailment properties.

The basic idea of Natural Logic goes all the way back to Aristotle's syllogisms, and in more modern times has attracted interest from scholars from generative semantics (Lakoff 1970) and those from formal and computational semantics (Fyodorov et al. 2000; Purdy 1991; Sánchez Valencia 1991; Zamansky et al. 2003). It has also been used to define controlled natural languages (McAllester and Givan 1992; Sukkarieh 2001). Recently, it has been revived in the context of automatically determining textual entailment (MacCartney and Manning 2009), see Section 6.2.

Natural Logic offers an interesting alternative to reasoning with first-order logic, because the problem of mapping syntactic structures to semantic representations doesn't exist, and it can model phenomena that go beyond the power of first-order logic, such as generalised quantifiers. The drawbacks of Natural Logic systems are dealing with context-sensitive expressions such as pronouns and ellipsis, and applying inferences to larger text fragments that go beyond single sentences.

## *5. Knowledge Issues*

### 5.1. THE QUEST FOR BACKGROUND KNOWLEDGE

Most of the natural language inference tasks that we described before require additional *background knowledge*. Without it, situations can occur where required inferences are simply not concluded, or where the generated models are not accurate. The complexity of background knowledge can range from extremely simple bits of knowledge to complicated sets of rules. For instance, simple background knowledge might state that a woman is a female person, and that the pronoun *he* refers to male entities, and that male and female entities form disjoint sets. More complicated rules might describe physical laws, such as *if X is contained in Y, and Y is contained in Z, then X is also contained in Z.* The issue of computing and using background knowledge is an important one and covers the following issues:

- How do we incorporate background knowledge into the inference tasks such as consistency and informativeness checking?
- To what extent can we rely on manually coded background knowledge and how can we automatically construct background knowledge in appropriate format from natural language resources?
- How can we limit the amount of required background knowledge in order to reach appropriate levels of efficiency in practical applications?

Before looking at these three questions in more detail, it is worth pointing out that background knowledge can come from different sources. As a guiding principle, it is often useful to distinguish knowledge into three classes (Blackburn and Bos 2005): lexical knowledge (the meaning of words), world knowledge (general facts about the world, such as in the CYC project (Lenat 1995)) and situational knowledge (facts that hold in the current situation, only relevant for systems embedded in practical applications such as robots). Here we will be mostly concerned with lexical knowledge.

In some logical frameworks, notably in description logic (Baader et al. 2002), a clear distinction is made between the background knowledge and the information on which the inference is about, called T-Box (short for the box containing the *theory*) and A-Box (short for the box containing the *assertion*), respectively. The questions of how to incorporate background knowledge is, when working in first-order logic, easy to answer. We can formulate knowledge as axioms of first-order logic, and supply them to the inference engines in addition to the inference problem itself. We have already demonstrated how to do this in Section 4.2.

## 5.2. COMPUTING BACKGROUND KNOWLEDGE

In this section, we are concerned with computing background knowledge for open-domain applications. Formulating appropriate lexical background knowledge is a difficult business even for restricted domains, and doing so for open domains requires the use of large-scale lexical resources. A good starting point for obtaining lexical knowledge is to look at electronic dictionaries such as WordNet (Fellbaum 1998) or Wikipedia for large-scale taxonomic knowledge (Suchanek et al. 2009), or NomLex (Meyers et al. 1998) for nominalisation patterns. Here we will consider NomLex and WordNet in more detail, and see how their lexical relations can be systematically translated as background knowledge formulated in first-order logic. Alternatively, one could consider automatically inducing semantic relations using large corpora. We will illustrate this idea as well.

### 5.2.1. Nominalisation Patterns

The linguistic phenomenon called *nominalisation* is basically the process of turning verbal phrases into nouns. An example illustrating nominalisation is the following, where the noun *eruption* is a nominalisation of the verb *to erupt*. Clearly, these phrases are semantically related – an expression with a prepositional phrase like 'the eruption of X' or a prenominal genitive expression 'X's eruption' entails that 'X erupted'. More generally speaking, there is a syntactic relationship between nominalisations and their verbal origins that can be exploited semantically (admission – to admit, blocking – to block, classification – to classify, destruction – to destroy, and so on). We demonstrate how to do so by analysing the following pair of sentences in first-order logic, using a neo-Davidsonian way to model events discussed in Section 2.1.3:

*The eruption of Mount Etna in 2008 created lava flows.*
$\exists x(\textsc{eruption}(x) \wedge \textsc{of}(x,\textsc{etna}) \wedge \textsc{in}(x,2008) \wedge$
$\quad \exists y(\textsc{lava-flow}(y) \wedge \exists e(\textsc{create}(e) \wedge \textsc{agent}(e,x) \wedge \textsc{theme}(e,y))))$

*Mount Etna erupted in 2008.*
$\exists e(\textsc{erupt}(e) \wedge \textsc{theme}(e,\textsc{etna}) \wedge \textsc{in}(e,2008))$

The first sentence entails the second one. But if we give the first-order translations to a theorem prover, and ask whether this is indeed the case, the prover obviously won't find a proof. Because what is missing is background knowledge that links the nominalisation with its verbal root. This piece of knowledge could be formulated in first-order logic as follows:

$\forall x \forall y(\textsc{eruption}(x) \wedge \textsc{of}(x,y) \leftrightarrow \textsc{erupt}(x) \wedge \textsc{theme}(x,y))$

Given that the syntactic relationship between a verb and its nominalisation appears to be irregular (there are various types of morphological transformations), it is important to have access to correct nominalisation patterns. The NomLex database contains such patterns for a large number of English nominalisations (Meyers et al. 1998). The patterns cover various nominalisation types and include verbal subcategorisation information of both the verb and the corresponding nominal form. The patterns are not formulated in first-order logic, but as syntactic patterns in feature structure notation. Here is a (slightly simplified) example of a NomLex entry:

```
(NOM :ORTH "eruption"
     :VERB "erupt"
     :NOM-TYPE ((VERB-NOM))
     :VERB-SUBJ ((N-N-MOD)
                 (DET-POSS)
                 (PP :PVAL ("by" "of")))
```

This is, of course, the entry for *eruption*, and it contains the verb associated with the nominalisation, a list of nominal positions where the verbal subject can be found (DET-POSS, for instance, says that X in 'X's eruption' would play the role of subject for the verb 'erupt'). It is clear that NomLex entries can be systematically translated into first-order axioms such as the one shown above, even though several issues such as disambiguation and thematic role labelling need to be taken into account. As a matter of fact, NomLex is a fine example of a lexical resource for formulating background knowledge rules, and it has been used to serve inference tasks in applications such as question answering (Bos 2006) and RTE (Bar-Haim et al. 2009).

Nevertheless, at this point one might ask why we would construct these axioms as part of the background knowledge rather than formulating it directly in the lexicon. A good question, which doesn't have a straightforward answer. On the one hand, if a nominalisation such as *eruption* is already semantically encoded as an event of the type 'erupt', decorated with the appropriate thematic roles, then the background knowledge associating the meaning of the noun with the meaning verb would be obsolete. This is a view that goes in the direction of frame semantics, and in fact the earlier mentioned FrameNet (Baker et al. 1998) would assign the same frame to the two different syntactic realisations in the examples above. On the other hand, one could leave the work at the lexical level simple, and only include the relevant axioms in the background knowledge when they are actually needed. Such a strategy will keep the set of supporting axioms as small as possible, with less work to be done at the lexical level.

### 5.2.2. Synonyms and Hyponyms

Given a vocabulary of non-logical symbols, we will also need to have background knowledge that tells us how these symbols are related to each other. Symbols might refer to the same concept and therefore have the same meaning (say RAPTOR and BIRD-OF-PREY), they might be disjoint concepts (for instance BUZZARD and KESTREL), or they might form subclasses of each other (KESTREL is a subclass of FALCON, and FALCON a subclass of RAPTOR, and so on). This ontological (or taxonomical) information can be derived from electronic dictionaries such as WordNet (Fellbaum 1998).

WordNet organises words into sets of synonyms, the *synsets* in WordNet parlance. One word can be part of more than one synset: the word *dog*, for instance, is part of the synsets:

{*dog*, *domestic dog*, *Canis familiaris*} and

{*hotdog*, *dog*, *wienerwurst*}.

We would then say that the word dog can be mapped to two senses: $dog_1$ (the animal sense) and $dog_2$ (the food sense). But the real added value of WordNet is that it interlinks all its synsets by semantic relations. One interesting relation, for our purpose of formulating background knowledge, is the hyponymy relation: $boxer_4$ and $poodle_1$ are hyponyms of the $dog_1$ sense, $dog_1$ is a hyponym of $mammal_1$, $mammal_1$ is a hyponym of $vertebrate_1$, and so on, and $dog_2$ is a hyponym of $sausage_1$. Why is this of interest to us? The hyponymy relations almost directly translate into background knowledge:

$$\forall x(\text{POODLE}_1(x) \rightarrow \text{DOG}_1(x))$$
$$\forall x(\text{BOXER}_4(x) \rightarrow \text{DOG}_1(x))$$
$$\forall x(\text{DOG}_1(x) \rightarrow \text{MAMMAL}_1(x))$$
$$\forall x(\text{MAMMAL}_1(x) \rightarrow \text{VERTEBRATE}_1(x))$$

The hyponym relation can give us also information about disjointness. We could stipulate that two senses with the same hypernym are disjoint. For instance, as $boxer_4$ and $poodle_1$are both (direct) hyponyms of $dog_1$, we could formulate this in the following background axiom:

$$\forall x(\text{POODLE}_1(x) \rightarrow \neg\text{BOXER}_4(x))$$

The use of WordNet relations is an attractive option for computing background knowledge but nevertheless, it comes at a cost. Most of the common words appear in more than one synset. In order to select the right synset, we need to pick the correct sense of the word, an incredibly hard problem. Recall from Section 3.2 that this process is called *WSD*, and it's one of the classic tasks of NLP (Jurafsky and Martin 2000).

A further concern with relying on WordNet is its lexical coverage: it contains almost all ordinary words, but no terms of specialised domains. However, several hand-crafted ontologies exist for biological and medical domains, which can serve the same purpose as WordNet does. For an introduction to ontology and population from text, the reader is referred to Cimiano (2006). Alternatively, a promising way is to compute hyponym relations directly from raw texts using pattern matching, an approach that we discuss in the next section.

### 5.2.3. Extracting Semantic Relations from Text

Instead of relying on manually crafted lexical resources, one could also calculate lexical knowledge on the basis of large corpora. An active area of research in computational linguistics is to derive instances of hyponymy (and other) relations from raw texts. The most straightforward of way of doing this is with the help of regular expressions, a method pioneered by Hearst (1992). To find hyponyms of *dog*, queries like

```
"dogs such as * and"
"such dogs as * and"
```

where the $*$ is a wildcard for any word, can be successfully fed into information retrieval engines and applied to search in large corpora, or even on the world wide web using ordinary search engines such as Google. In fact, using Google on the above pattern generates the hyponyms *Terriers*, *collies*, *Rottweilers* and *Chihuahuas* as first hits, beautifully demonstrating the usefulness of this technique.

Harvesting hyponym relations from corpora with such methods is an interesting approach, but often additional measures have to be taken into account to separate the wheat from the chaff, because almost any pattern can produce false positives. A nice example illustrating this problem is using the search pattern above on the sentence 'Serovars that infect dogs such as grippotyphosa and pomona …', which would wrongly classifying *grippotyphosa* as a hyponym of dog! Usually, a statistical test (based on frequency) is employed in these methods to decide whether an instance found in a corpus is reliable or not (Pantel and Pennacchiotti 2006; Tjong Kim Sang 2007). Alternatively, one could explore methods that don't rely on surface patterns, but instead take advantage of syntactic analyses (using a robust parser) or even semantic representations (using a wide-coverage semantic processor) to get rid of noise that would be unavoidably generated by using simple surface patterns (Tjong Kim Sang and Hofmann 2009).

Besides harvesting hyponyms, the pattern-based approach for finding semantic relations has also been exploited for meronymy (Girju et al. 2006) and various relations between verbs (Chklovski and Pantel 2004). One difficulty that the pattern-based approach to finding semantic relations in raw text faces is that it doesn't say anything about the senses of words in the extracted patterns. To stay in the spirit of the example above, we could also find (mistakenly) hyponyms for the *sausage* sense of dog. The problem of correctly identifying the appropriate word sense of each new word added to the taxonomy is approached by Snow et al. (2006).

Promising, alternative approaches are unsupervised methods for detecting inference rules, such as paraphrases (Lin and Pantel 2001), synonyms of rare words (Curran 2004) and analogies (Turney 2006). Algorithms in this paradigm of research are based on the *Distributional Hypothesis* of Harris (1954), which simply says that words that occur in the same contexts, are likely to have a similar meaning.

### 5.3. SELECTING APPROPRIATE BACKGROUND KNOWLEDGE

The third and final problem related to background knowledge that we discuss here is the selection of background knowledge. Let's illustrate this problem by using WordNet as the main source of lexical knowledge. A naive computational semanticist would neatly translate all the hyponym relations of WordNet into first-order axioms. However, even the state-of-the-art theorem provers for first-order logic would be unable to work on such a large amount of information: there are about 115,000 synsets in WordNet! Obviously, we should only give the lexical information to the theorem prover that is required for a proof – but how do we know what constitutes this information?

Perhaps, at first sight, this looks like a trivial problem. Aren't we interested in performing inferences for certain expressions of natural language, and can't we just calculate the lexical knowledge only for those terms appearing in the expression of our interest? For instance, to find out whether a sentence is informative with respect to a discourse, why not calculate hyponym relations *only* for the nouns appearing in the sentence and discourse? A sensible idea, yet the problem is that everything in the world seems to relate, directly or indirectly, to each other. If we talk about dogs, then since dogs are animals, do we need to incorporate knowledge about animals? If we introduce knowledge about animals, do we need to state general facts about animals (for instance that they are living creatures and they eat and sleep). But if we do that, do we need to supply background knowledge about eating and sleeping habits?

On the one hand, it seems that just supplying background knowledge for primary terms of the inference problem is not enough. On the other hand, for practical reasons, we need to stop at some point generating background knowledge, but where? One way to deal (partially) with this problem is working with subsets of WordNet (or other ontologies) that are computed on the fly depending on the non-logical symbols contained in the inference problem, as is done by Bos (2005), and taken further by Wotzlaw (2010) by including concepts from Wikipedia. Here the basic idea of using WordNet as the backbone for producing lexical knowledge is used, but with a twist: only a portion of WordNet relevant for a certain natural language expression is used, by computing the smallest closure of hyponym relations spanning a set of terms. This is, however, just one step towards approaching the real problem, and the field would benefit from further methods and techniques tackling this issue.

## *6. Evaluation in Computational Semantics*

The development of broad-coverage, open-domain systems for semantic interpretation, as discussed in Section 3.3 is an interesting one but also gives rise to a new challenge: that of measuring the accuracy of the meaning representations they produce. Most of the systems report their success simply by numbers on coverage: the ratio of sentences (or texts) for which they are able to produce semantic representations. Such figures, obviously, only say something about quantity, and nothing about the quality of the systems' output.

All of these approaches have in common that they claim to represent natural language meaning, but due to the use of different theoretical frameworks the level and depth of semantic analysis show how a high variety in representational detail. This is understandable but unfortunate, because it makes it harder to compare output and evaluate such systems.

Therefore, a crucial question in the future development of computational semantics is how we shall go about in evaluating the semantic adequacy of meaning representations, and the correctness of the automatically drawn natural language inferences. The 'Glass-Box' method, comparing system output with a gold standard, would be effective for assessing semantic adequacy, but has various drawbacks. Alternatively, one could take 'Black-Box' approaches to semantic evaluation. Below we will discuss two such methods: the task of RTE and the task of textual model checking.

### 6.1. QUALITY OF SEMANTIC REPRESENTATIONS

An obvious way of assessing the quality of a meaning interpretation component is to compare the semantic representations that a system produces with a gold standard annotation. After all, that's what researchers in NLP do when evaluating part-of-speech tagging, chunking, named entity recognition and syntactic parsing. Allen et al. (2008) introduce an evaluation metric for semantic representations coded in graph formats. This metric, defined in terms of precision and recall, is computed on the basis of the maximum score produced by any node/edge alignment from the gold semantic representation to the one produced by the system.

Unavoidably, several practical questions emerge. What exactly should such a gold standard for meaning representations look like? What ingredients should be part of an adequate semantic representation? Should it follow a particular (formal) theory of semantics, or rather take an independent stance – if that indeed is a possibility? What semantic phenomena should it aim to cover, and in what level of detail should they be analysed? Even if comparing to a gold standard would be an effective methodology for assessing semantic adequacy, annotating text with semantic representations is an immense task, with many choices to make (Bos 2008b). To date no large annotated corpus integrating a wide inventory of semantic phenomena exist.

### 6.2. RECOGNISING TEXTUAL ENTAILMENT

The task of automatically recognising entailment relations for pairs of text was first proposed in the FRACAS project (Cooper et al. 1996), comprising a test suite with about 350 examples, grouped into linguistic and semantic phenomena and presented as a list of textual premises followed by a yes/no-question and the correct answer (*yes*, *no* or *don't know*). As a shared task, RTE became popular when it was organised in the context of the PASCAL challenges (Bar-Haim et al. 2006; Dagan et al. 2006; Sekine et al. 2007), even though the suggestion had been made earlier by Monz and de Rijke (2001). The PASCAL RTE data is similar to the FRACAS test suite examples, but based on real data (rather than artificially constructed examples), and are not grouped or marked for any specific semantic phenomenon. The PASCAL data set consists of text-hypothesis pairs, and are marked as either true (the hypothesis follows from the text) or false (hypothesis doesn't follow from the text).

At first sight, the RTE task is an attractive method for evaluating semantics because it is theory neutral and works on open-domain data. Yet, various issues arise. One problem with the open domain PASCAL RTE examples is the difficulty of isolating the semantic interpretation task from the task of acquiring the relevant background knowledge (Zaenen et al. 2005). In contrast, the FRACAS test suite was deliberately constructed to limit the amount of additional background knowledge to make the required inferences.

A second problem is that the RTE task doesn't assess each component in an textual entailment system separately. This makes it harder to evaluate the semantic aspects only. For instance, an RTE system could have a splendid inference system (perhaps based on first-order logic) but still get a low score when ran on the RTE test suites because it has a bad performing named entity recogniser, parser or WSD component.

### 6.3. TEXTUAL MODEL CHECKING

An alternative black-box evaluation method is to perform the evaluation of a semantic component by *model checking*. Simply put, an NLP system is given a sentence (or text) and a model (a model in the sense as used in Section 2) and is asked whether the sentence is true or false (or unknown) in the given model (Bos 2008b).

Like the RTE task, this method has the advantage that it is reasonably independent of the semantic representation language, even though it presupposes a common vocabulary of non-logical symbols and is therefore not completely theory neutral. The model checking method seems perhaps far away from practical applications, but in fact can be seen as a more general form of testing natural language query interfaces to relational databases, as Tang and Mooney (2000) show for a geographical database.

It remains unclear how hard it is to construct large test suites of natural, open domain examples. However, it seems that, once there is an initial set of examples of texts and model pairs, the amount of training data can be rather straightforwardly extended by inducing (small) variants of the existing models. An additional advantage is that it seems relatively easy to construct both positive and negative examples, facilitating the creation of balanced test suites.

## 7. *Outlook*

Looking at the current developments in the field, one can only conclude that it is an exciting time for computational semantics, not least because of the rise of implementations of robust wide-coverage systems for semantic interpretation, but also because of ongoing theoretical work concerning alternative representations and inference methods. These are often stimulated by interdisciplinary interest outside computational semantics, coming from the areas of knowledge representation, automated deduction, as well as that of formal semantics. Moreover, recently there have been many practical considerations focusing on the organisation of evaluation campaigns for testing semantic adequacy.

But let's shift our attention from now to the future, and concentrate on what the next big problems to be solved in computational semantics really are. We can't answer this question without indulging ourselves in some serious speculation. Nevertheless, we think they can be summarised by three topics: machine learning, knowledge resources and scaling inference.

### 7.1. MACHINE LEARNING

Statistical approaches have pushed computational syntax forward in the past 10 years, and they could have a similar impact on computational semantics. It hasn't happened yet mainly because large gold standard annotated corpora, integrating various deep semantic phenomena rather than focusing on one, aren't yet available. One could try to create such corpora by merging existing resources for single semantic phenomena (Pustejovsky et al. 2005). Alternatively, one could build a large resource from scratch, but given the enormous effort this requires, it is unclear when a corpus large enough to create high-performance statistical models will be available, or indeed, if its realisation is feasible at all.

### 7.2. KNOWLEDGE RESOURCES

Lexical knowledge is scattered over various resources and therefore difficult to integrate in systems. The problem of selecting appropriate background knowledge is further hindered by the large number of senses attributed to single word forms. Developing an infrastructure for systematically combining lexical knowledge with general world knowledge, without falling prey in a web of complexity would certainly benefit computational semantics. There are various ongoing initiatives on the world wide web trying to achieve this. Large-scale common sense knowledge bases, such as CYC (Lenat 1995), have been, somewhat surprisingly, only sporadically used in natural language interpretation systems, but deserve further exploitation.

### 7.3. SCALING INFERENCE

There are new developments in areas related to computational semantics that could have a high impact on the field itself. The recent progress in the area of Answer Set Programming (Gelfond 2008) is largely unexplored territory by computational semanticists, but offers interesting alternatives to first-order inference. Current trends in first-order theorem proving aim to include built-in theories such as transitive relations, ordering relations, non-standard quantifiers and arithmetic (Voronkov 2003); this is an interesting direction as it gives opportunities to scale first-order inference tasks to natural language phenomena comprising plurals, modalities and tense.

## Short Biography

Johan Bos got his doctoral degree from the University of the Saarland, Germany, and held postdoc positions at the University of Edinburgh, UK, and the "La Sapienza" University of Rome, Italy. He is currently Professor of Computational Semantics at the University of Groningen, the Netherlands, holding a prestigious endowed-chair position. His research interests cover all aspects of semantics in linguistic theory and natural language processing. He is co-author of the first textbook devoted to computational semantics, "Representation and Inference for Natural Language". He is also president of SIGSEM, the special interest group on computational semantics of the ACL, the association for computational linguistics.

## Notes

* Correspondence address: Johan Bos, Department of Computer Sciences, University of Groningen, Postbus 716, 9700 AS Groningen, Netherlands. E-mail: Johan.Bos@rug.nl

[1] We are ignoring function symbols in this article. A first-order language with function symbols allows one to use structured terms. This can be convenient, but is usually not necessary.
[2] This analysis of proper names is, of course, an oversimplification. Names can have complex structure, and can also exhibit anaphoric behaviour.
[3] However, in the extended fragment of DRT of Kamp and Reyle, which goes beyond the expressive power of first-order logic, so-called duplex conditions are introduced to represent explicit quantifiers.
[4] But what is reasonably small? To give an example, for the RTE-3 textual entailment test suite of the PASCAL shared task (see Section 6.2), the Nutcracker system (Bos and Markert 2005) computed as most frequent domain size 14 for a T–H pair (with a maximum domain size of 38). The model builders used by Nutcracker (Paradox and Mace) were able to compute such models in a couple of seconds for roughly 90% of the cases. However, bear in mind that these are simple models, without treating complex phenomena such as plurals and tense.

## Works Cited

Akhmatova, E., and D. Mollá. 2006. Recognizing textual entailment via atomic propositions. MLCW 2005. Vol. 3944 of Lecture Notes in Computer Science, ed. by J. Q. Candela, I. Dagan, B. Magnini and F. d'Alché-Buc, 385–403. Berlin/Heidelberg: Springer.

Allen, J. F., M. Swift, and W. de Beaumont. 2008. Deep semantic analysis of text. Semantics in text processing. STEP 2008 Conference Proceedings. Vol. 1 of Research in Computational Semantics, ed. by J. Bos and R. Delmonte, 343–54. London: College Publications.

Asher, N. 1993. Reference to abstract objects in discourse. Dordrecht: Kluwer Academic Publishers.

Asudeh, A., and R. Crouch. 2002. Glue semantics for HPSG. Proceedings of the 8th International HPSG Conference, ed. by van F. Eynde, L. Hellan, and D. Beermann, 1–19. Stanford, CA: CSLI Publications.

Baader, F., D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider (eds). 2002. The description logic handbook. Cambridge: Cambridge University Press.

Baker, C. F., C. J. Fillmore, and J. B. Lowe. 1998. The Berkeley FrameNet project. 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics. Proceedings of the Conference, 86–90. Montreal, QC, Canada: Université de Montréal.

Balduccini, M., C. Baral, and Y. Lierler. 2008. Knowledge representation and question answering. Handbook of knowledge representation, ed. by V. Lifschitz, F. van Harmelen, and B. Porter, 779–819. San Diego: Elsevier.

Bar-Haim, R., I. Dagan, B. Dolan, L. Ferro, and D. Giampiccolo. 2006. The second pascal recognising textual entailment challenge. Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, ed. by B. Magnini and I. Dagan, 1–9. Venice, Italy.

——, J. Berant, and I. Dagan. 2009. A compact forest for scalable inference over entailment and paraphrase rules. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, ed. by P. Koehn and R. Mihalcea, 1056–65. Singapore: World Scientific Publishing Co Pte Ltd.

Baral, C., J. Dzifcak, and T. C. Son. 2008. Using answer set programming and lambda calculus to characterize natural language sentences with normatives and exceptions. Proceedings of the 23rd National Conference on Artificial Intelligence, Vol. 2, ed. by A. Cohn, 818–23. Chicago: AAAI Press.

Bartsch, R. 1973. The semantics and syntax of number and numbers. Syntax and Semantics, ed. by J. P. Kimball, 51–93. New York: Seminar Press, Inc.

Barwise, J., and R. Cooper. 1981. Generalized quantifiers and natural language. Linguistics and Philosophy 4. 159–219.

Baumgartner, P., U. Furbach, and B. Pelzer. 2007. Hyper tableaux with equality. Automated Deduction – CADE-21, 21st International Conference on Automated Deduction. Vol. 4603. of Lecture Notes in Computer Science, ed. by F. Pfenning, 492–507. Berlin: Springer.

Blackburn, P., and J. Bos. 2003. Computational semantics. Theoria 18(46). 27–45.

——, and ——. 2005. Representation and inference for natural language. A first course in computational semantics. Stanford: CSLI.

——, ——, M. Kohlhase, and H. de Nivelle. 2001. Inference and computational semantics. Computing meaning Vol.2, ed. by H. Bunt, R. Muskens, and E. Thijsse, 11–28. Dordrecht: Kluwer.

Bos, J. 1996. Predicate logic unplugged. Proceedings of the Tenth Amsterdam Colloquium, ed. by P. Dekker and M. Stokhof, 133–43. Amsterdam: ILLC/Department of Philosophy, University of Amsterdam.

——. 2003. Implementing the binding and accommodation theory for anaphora resolution and presupposition projection. Computational Linguistics 29(2). 179–210.

——. 2005. Towards wide-coverage semantic interpretation. Proceedings of Sixth International Workshop on Computational Semantics IWCS-6, ed. by H. Bunt, J. Geertzen and E. Thijsse, 42–53. Tilburg: University of Tilburg.

——. 2006. The ''La Sapienza'' question answering system at TREC 2006. Proceeding of the Fifteenth Text RETrieval Conference, TREC-2006, ed. by E. Voorhees, and L. P. Buckland, 797–803. Gaithersburg, MD: NIST.

——. 2008a. Introduction to the shared task on comparing semantic representations. Semantics in text processing. STEP 2008 Conference Proceedings. Vol. 1 of Research in Computational Semantics, ed. by J. Bos and R. Delmonte, 257–61. London: College Publications.

——. 2008b. Let's not argue about semantics. Proceedings of the 6th Language Resources and Evaluation Conference (LREC 2008), ed. by N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odjik, S. Piperidis and D. Tapias, 2835–40. Marrakech, Morocco: European Language Resources Association (ELRA).

——. 2008c. Wide-coverage semantic analysis with boxer. Semantics in Text Processing. STEP 2008 Conference Proceedings. Vol. 1 of Research in Computational Semantics, ed. by J. Bos and R. Delmonte, 277–86. London: College Publications.

——, and K. Markert. 2005. Recognising textual entailment with logical inference. Proceedings of the Conference on Empirical Methods in Natural Language Processing, 628–35. Vancouver.

——, and M. Gabsdil. 2000. First-order inference and the interpretation of questions and answers. Proceedings of Götalog 2000, Fourth Workshop on the Semantics and Pragmatics of Dialogue, ed. by M. Poesio and D. Traum, 43–50. Gothenburg: University of Gothenburg. Gothenburg Papers in Computational Linguistics 00-5.

——, and S. Pulman (eds). 2011. Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011). Oxford, UK.

——, and T. Oka. 2007. Meaningful conversation with mobile robots. Advanced Robotics 21(1–2). 209–32.

Brachman, R., and H. Levesque. 2004. Knowledge representation and reasoning. San Francisco, CA: Morgan Kaufmann Publishers Inc.

Butler, A., Y. Miyao, K. Yoshimoto, and J. Tsujii. 2010. A constrained semantics for parsed english. Proceedings of the Sixteenth Annual Meeting of the Association of Natural Language Processing (NLP 2010), 836–9. Tokyo.

Butt, M., H. Dyvik, T. H. King, H. Masuichi, and C. Rohrer. 2002. The parallel grammar project. COLING-02 workshop on Grammar Engineering and Evaluation, 1–7. Morristown, NJ: Association for Computational Linguistics.

Carter, D. 1987. Interpreting anaphors in natural language texts. Chichester, UK: Ellis Horwood.

Chamberlain, J., M. Poesio, and U. Kruschwitz. 2008. Addressing the resource bottleneck to create large-scale annotated texts. Semantics in Text Processing. STEP 2008 Conference Proceedings. Vol. 1 of Research in Computational Semantics, ed. by J. Bos and R. Delmonte, 375–80. London: College Publications.

Chaves, R. P. 2007. Dynamic model checking for discourse representation structures with pluralities. 7th International Workshop on Computational Semantics, ed. by J. Geertzen, E. Thijsse, H. Bunt, and A. Schiffrin, 28–40. The Netherlands: Tilburg University.

Chklovski, T., and P. Pantel. 2004. Verbocean: mining the web for fine-grained semantic verb relations. Proceedings of EMNLP 2004, ed. by D. Lin, and D. Wu, 33–40. Barcelona, Spain: ACL.

Church, A. 1940. A formulation of a simple theory of types. Journal of Symbolic Logic 5(2). 56–68.

Cimiano, P. 2003. Building models for bridges. ICoS-4, Inference in Computational Semantics, Workshop Proceedings, ed. by P. Blackburn, and J. Bos, 57–71. Nancy, France.

——. 2006. Ontology learning and population from text: algorithms, evaluation and applications. Berlin: Springer.

Claessen, K., and N. Sörensson. 2003. New techniques that improve mace-style model finding. Model Computation – Principles, Algorithms, Applications (Cade-19 Workshop), ed. by P. Baumgartner and C. Fermüller, 11–27. Miami, FL.

Cooper, R. 1983. Quantification and syntactic theory. Dordrecht: Reidel.

——, D. Crouch, J. Van Eijck, C. Fox, J. Van Genabith, J. Jaspars, H. Kamp, M. Pinkal, D. Milward, M. Poesio, and S. Pulman. 1996. Using the framework. Tech. rep., FraCaS: a framework for computational semantics, fraCaS deliverable D16.

Copestake, A., and D. Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000), ed. by M. Gavrilidou, G. Crayannis, S. Markantonatu, S. Piperidis and G. Stainhaouer, 591–600. Athens, Greece: European Language Resources Association (ELRA).

——, ——, I. Sag, and C. Pollard. 2005. Minimal recursion semantics: an introduction. Journal of Research on Language and Computation 3(2–3). 281–332.

Crouch, D., and T. Holloway King. 2006. Semantics via F-structure rewriting. Proceedings of the LFG06 Conference, ed. by M. Butt and T. Holloway King, 1–20. Konstanz: CSLI publications.

Curran, J., S. Clark, and J. Bos. 2007. Linguistically motivated large-scale NLP with C&C and boxer. Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions, ed. by S. Ananiadou, 33–36. Prague, Czech Republic: Association for Computational Linguistics.

Curran, J. R. 2004. From distributional to semantic similarity. University of Edinburgh, Ph.D. thesis.

Dagan, I., O. Glickman, and B. Magnini. 2006. The pascal recognising textual entailment challenge. Lecture notes in computer science, Vol. 3944, 177–90. Berlin: Springer.

Dalrymple, M., S. M. Shieber, and F. C. Pereira 1991. Ellipsis and higher-order unification. Linguistics and Philosophy 14. 399–452.

Dowty, D. 1989. On the semantic content of the notion ''thematic role''. Properties, types, and meanings, Vol. 2, ed. by G. Chierchia, B. Partee and R. Turner, 69–129. Dordrecht: Kluwer.

——, R. Wall, and S. Peters. 1981. Introduction to montague semantics. Dordrecht: Reidel.

Egg, M., J. Niehren, P. Ruhrberg, and F. Xu. 1998. Constraints over lambda-structures in semantic underspecification. 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics. Proceedings of the Conference, 353–9. Montreal, QC, Canada: Université de Montréal.

van Eijck, J., and H. Kamp. 1997. Representing discourse in context. Handbook of logic and language, ed. by J. van Benthem, and A. ter Meulen, 179–240. Amsterdam: Elsevier, MIT.

Erk, K., and S. Pado. 2006. Shalmaneser – a toolchain for shallow semantic parsing. 5th Edition of the International Conference on Language Resources and Evaluation, 527–32. Genoa, Italy.

Fellbaum, C. (ed.). 1998. WordNet. An electronic lexical database. Cambridge, MA: The MIT Press.

Franconi, E. 1993. A treatment of plurals and plural quantifications based on a theory of collections. Minds and machines, ed. by S. C. Shapiro, 453–74. Dordrecht: Kluwer Academic Publishers.

——. 2002. Description logics for natural language processing. The description logic handbook, Ch. 15, ed. by F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, 450–60. Cambridge: Cambridge University Press

Furbach, U., I. Glöckner, and B. Pelzer. 2010. An application of automated reasoning in natural language question answering. AI Communications 23(2–3). 241–65.

Fyodorov, Y., Y. Winter, and N. Francez. 2000. A natural logic inference system. ICoS-2, Inference in Computational Semantics, Workshop Proceedings, ed. by J. Bos and M. Kohlhase, 72–88. Germany: Schloss Dagstuhl.

Gamut, L. 1991. Logic, language, and meaning. Volume II. Intensional logic and logical grammar. London: The University of Chicago Press.

Gardent, C., and B. Webber. 2001. Towards the use of automated reasoning in discourse disambiguation. Journal of Logic Language and Information 10(4). 487–509.

——, and K. Konrad. 2000. Interpreting definites using model generation. Journal of Language and Computation 1(2). 193–209.

Gelfond, M. 2008. Answer sets. Handbook of knowledge representation, ed. by V. Lifschitz, F. van Harmelen, and B. Porter. 285–316. San Diego: Elsevier.

van Genabith, J., A. Frank, and R. Crouch. 1999. Glue, underspecification and translation. IWCS-3, Third International Workshop for Computational Semantics, ed. by H. C. Bunt, and E. G. C. Thijse, 265–79. Tilburg: University of Tilburg.

Girju, R., A. Badulescu, and D. Moldovan. 2006. Automatic discovery of part-whole relations. Computational Linguistics 32(1). 83–135.

Hardt, D. 1997. An empirical approach to VP ellipsis. Computational Linguistics 23(4). 525–41.

Harris, Z. 1954. Distributional structure. Word 10(23). 146–62.

Hearst, M. A. 1992. Automatic acquisition of hyponyms from large text corpora. Proc. of COLING-92, 539–45. Nantes.

Hintikka, J. 1973. Quantifiers vs. quantification theory. Dialectica 27(3–4). 329–58.

Hobbs, J. R., M. E. Stickel, D. Appelt, and P. Martin. 1990. Interpretation as abduction. Menlo Park, CA: Tech. Rep. 499, AI Center, SRI International.

Hoeksema, J. 1983. Plurality and conjunction. Studies in Modeltheoretic Semantics, ed. by A. G. ter Meulen, 63–83. Dordrecht: FLORIS.

Jurafsky, D., and J. H. Martin. 2000. Speech and language processing. An introduction to natural language processing, computational linguistics, and speech recognition. New Jersey: Prentice Hall.

Kamp, H., and U. Reyle. 1993. From discourse to logic; an introduction to modeltheoretic semantics of natural language, formal logic and DRT. Dordrecht: Kluwer.

Kehler, A. 1993. A discourse copying algorithm for ellipsis and anaphora resolution. EACL, 203–12. Utrecht: OTS, University of Utrecht.

Keller, W. R. 1988. Nested cooper storage: the proper treatment of quantification in ordinary noun phrases. Natural language parsing and linguistic theories, ed. by U. Reyle and C. Rohrer, 432–47. Dordrecht: Reidel.

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

Koller, A., K. Mehlhorn, and J. Niehren. 2000. A polynomial-time fragment of dominance constraints. Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics, 376–83. Hong Kong: Association for Computational Linguistics.

——, R. Debusmann, M. Gabsdil, and Striegnitz, K. 2004. Put my galakmid coin into the dispenser and kick it: computational linguistics and theorem proving in a computer game. Journal of Logic, Language, and Information, 13. 187–206.

——, and S. Thater. 2010. Computing weakest readings. Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10), Uppsala.

Lakoff, G. 1970. Linguistics and natural logic. Synthese 22(1–2). 151–271.

Leidner, J. L. 2008. Toponym resolution in text: annotation, evaluation and applications of spatial grounding of place names. Boca Raton, FL: Universal Press.

Lenat, D. 1995. Cyc: a large-scale investment in knowledge infrastructure. Communications of the ACM 38. 33–38.

Lierler, Y., and G. Görz. 2006. Model generation for generalized quantifiers via answer set programming. Proceedings of KONVENS 2006, ed. by M. Butt, 101–6. Konstanz.

Lin, D., and P. Pantel. 2001. DIRT—discovery of inference rules from text. Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 323–8. San Francisco, CA.

Link, G. 1983. The logical analysis of plurals and mass terms: a lattice-theoretical approach. Meaning, Use, and Interpretation of Language, ed. by R. Bäuerle, C. Schwarze, and A. von Stechow, 302–23. Berlin: Mouton.

Ludwig, B., G. Görz, and H. Niemann. 2000. An inference-based approach to the interpretation of discourse. Journal of Language and Computation 1(2). 261–76.

MacCartney, B., and C. D. Manning. 2009. An extended model of natural logic. Proceedings of the Eight International Conference on Computational Semantics, ed. by H. Bunt, V. Petukhova, and S. Wubben, 140–56. Tilburg: University of Tilburg.

Marcus, M., B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English. The Penn Treebank. Computational Linguistics 19(2). 313–30.

Markert, K., and M. Nissim. 2009. Data and models for metonymy resolution. Language Resources and Evaluation 43. 123–38.

Matthews, P. H. 1997. The concise Oxford dictionary of linguistics. Oxford: Oxford University Press.

McAllester, D., and R. Givan. 1992. Natural language syntax and first order inference. Artificial Intelligence 56. 1–20.

McCune, W. 1998. Automatic proofs and counterexamples for some ortholattice identities. Information Processing Letters 65(6). 285–91.

——, and R. Padmanabhan. 1996. Automated deduction in equational logic and cubic curves. No. 1095 in Lecture Notes in Computer Science (AI subseries). Berlin: Springer-Verlag.

Meyers, A., C. Macleod, R. Yangarber, R. Grishman, L. Barrett, and R. Reeves. 1998. Using nomlex to produce nominalization patterns for information extraction. Coling-ACL98 workshop Proceedings, The Computational Treatment of Nominals, 25–32. Montreal, Canada: Association for Computational Linguistics.

Mitkov, R. 2002. Anaphora resolution. London: Longman.

Moldovan, D. I., C. Clark, S. M. Harabagiu, and D. Hodges. 2007. Cogex: a semantically and contextually enriched logic prover for question answering. Journal of Applied Logic 5(1). 49–69.

Montague, R. 1973. The proper treatment of quantification in ordinary English. Approaches to natural language, ed. by J. Hintikka, J. Moravcsik, and P. Suppes, 221–42. Dordrecht: Reidel.

Monz, C., and M. de Rijke. 2001. Light-weight entailment checking for computational semantics. Workshop Proceedings ICoS-3, ed. by P. Blackburn, and M. Kohlhase, 59–72. Siena, Italy.

Moore, R. C. 1986. Problems in logical form. Natural language processing, ed. by B. J. Grosz, K. Sparck Jones, and B. L. Webber, 285–92. Los Altos, CA: Kaufmann.

Muskens, R. 1996. Combining montague semantics and discourse representation. Linguistics and Philosophy 19. 143–86.

Navigli, R. 2009. Word sense disambiguation: a survey. ACM Computing Surveys 41(2). 1–69.

Nielsen, L. A. 2005. A corpus-based study of verb phrase ellipsis identification and resolution. London: King's College Ph.D. thesis.

Palmer, M., P. Kingsbury, and D. Gildea. 2005. The proposition bank: an annotated corpus of semantic roles. Computational Linguistics 31(1). 71–106.

Pantel, P., and M. Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, ed. by N. Calzolari, C. Cardie and P. Isabelle, 113–20. Morristown, NJ: Association for Computational Linguistics.

Parsons, T. 1990. Events in the semantics of English: a study in subatomic semantics. Cambridge, MA: The MIT Press.

Pelletier, F., G. Sutcliffe, and C. Suttner. 2002. The development of CASC. AI Communications 15(2–3). 79–90.

Pelletier, F. J., and N. Asher. 1997. Generics and defaults. Handbook of logic and language, Ch. 20. J. Van Benthem, and A. Ter Meulen, 1125–77. Amsterdam: Elsevier, MIT.

Pratt-Hartmann, I. 2003. A two-variable fragment of english. Journal of Logic, Language and Information 12(1). 13–45.

——. 2004. Fragments of language. Journal of Logic, Language and Information 13(2). 207–23.

Purdy, W. C. 1991. A logic for natural language. Notre Dame Journal of Formal Logic 32(3). 409–25.

Pustejovsky, J. 1995. The generative Lexicon. Cambridge, MA: The MIT Press.

——, A. Meyers, M. Palmer, and M. Poesio. 2005. Merging PropBank, NomBank, TimeBank, Penn Discourse Treebank and Coreference. CorpusAnno '05: Proceedings of the Workshop on Frontiers in Corpus Annotations II, 5–12. Ann Arbor, MI: Association for Computational Linguistics.

Ramsay, A., and H. Seville. 2000. Models and discourse models. Language and Computation 1(2). 167–81.

Reiter, R. 1980. A logic for default reasoning. Artificial Intelligence 13(1–2). 81–132.

Reyle, U. 1993. Dealing with ambiguities by underspecification: construction, representation and deduction. Journal of Semantics 10. 123–79.

Riazanov, A., and A. Voronkov. 2002. The design and implementation of vampire. AI Communications 15(2–3). 91–110.

Robaldo, L. 2007. Dependency tree semantics. University of Turin, Ph.D. thesis.

Rodrigo, Á., A. Peñas, E. Hovy, and E. Pianta. 2010. Question answering for machine reading evaluation. Proceedings of CLEF 2010 Workshop on Question Answering in Multiple Languages (MLQA'10). Padua.

Sánchez Valencia, V. 1991. Studies on natural logic and categorial grammar, University of Amsterdam Ph.D. thesis.

van der Sandt, R. 1992. Presupposition projection as anaphora resolution. Journal of Semantics 9. 333–77.

Sato, M., D. Bekki, Y. Miyao, and J. Tsujii. 2006. Translating HPSG-style outputs of a robust parser into typed dynamic logic. Proceedings of the COLING/ACL on Main Conference Poster Sessions, 707–14. Morristown, NJ.

Scha, R. J. 1984. Distributive, collective and cumulative quantification. Truth, interpretation and information, ed. by J. Groenendijk, T. M. Janssen, M. Stokhof, 131–58. Dordrecht: FORIS.

Sekine, S., K. Inui, I. Dagan, B. Dolan, D. Giampiccolo, and B. Magnini (eds). June 2007. Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing. Prague: Association for Computational Linguistics.

Sher, G. 1990. Ways of branching quantifiers. Linguistics and Philosophy 13. 393–422.

Shutova, E. 2010. Models of metaphor in NLP. Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10), 688–97. Uppsala.

Snow, R., D. Jurafsky, and A. Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, 801–8. Sydney, Australia.

Soon, W. M., D. Chung Yong Lim, and H. T. Ng. 2001. A machine learning approach to coreference resolution of noun phrases. Computational Linguistics 27(4). 521–44.

Suchanek, F. M., M. Sozio, and G. Weikum. 2009. Sofie: a self-organizing framework for information extraction. 18th International World Wide Web conference (WWW 2009), 631–40. Madrid, Spain.

Sukkarieh, J. 2001. Quasi-NL knowledge representation for structurally-based inferences. ICoS-3, Inference in Computational Semantics, Workshop Proceedings, ed. by P. Blackburn and M. Kohlhase, 131–9. Siena, Italy.

Sutcliffe, G., and C. Suttner. 2006. The state of CASC. AI Communications 19(1). 35–48.

Tang, L. R., and R. J. Mooney. 2000. Automated construction of database interfaces: integrating statistical and relational learning for semantic parsing. Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 133–41. Hong Kong.

Tarski, A. 1956. The concept of truth in formalized languages. Logic, semantics, metamathematics, ed. by J. H. Woodger, 152–278. Oxford: Oxford University Press.

Tjong Kim Sang, E. 2007. Extracting hypernym Pairs from the web. ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, 165–8. Prague, Czech Republic.

——, and K. Hofmann. 2009. Lexical patterns or dependency patterns: which is better for hypernym extraction? Proceedings of CoNLL-2009, 174–82. Boulder, CO.

Turney, P. D. 2006. Similarity of semantic relations. Computational Linguistics 32(3). 379–416.

Voronkov, A. 2003. Automated reasoning: past story and new trends. Proceedings of the 18th International Joint Conference on Artificial Intelligence, ed. by G. Gottlob and T. Walsh, 1607–12. San Francisco, CA: Morgan Kaufmann Publishers Inc.

Weidenbach, C., B. Afshordel, U. Brahm, C. Cohrs, T. Engel, E. Keen, C. Theobalt, and D. Topić. 1999. System description: Spass version 1.0.0. 16th International Conference on Automated Deduction, CADE-16. Vol. 1632 of LNAI, ed. by H. Ganzinger, 314–8. Berlin: Springer-Verlag.

Wotzlaw, A. 2010. Towards better ontological support for recognizing textual entailment. The 17th International Conference on Knowledge Engineering and Knowledge Management, 316–30. Lisbon, Portugal.

Zaenen, A., L. Karttunen, and R. Crouch. June 2005. Local textual inference: can it be defined or circumscribed? Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, ed. by B. Dolan and I. Dagan, 31–36. Ann Arbor, MI: Association for Computational Linguistics.

Zamansky, A., N. Francez, and Y. Winter. 2003. A 'Natural Logic' inference system using normalisation. ICoS-4, Inference in Computational Semantics, Workshop Proceedings, ed. by P. Blackburn, and J. Bos, 197–232. Nancy, France.