Computational Semantics

Patrick BLACKBURN, Johan BOS

ABSTRACT: In this article we discuss what constitutes a good choice of semantic representation, compare different approaches of constructing semantic representations for fragments of natural language, and give an overview of recent methods for employing inference engines for natural language understanding tasks.

Keywords: Formal semantics, computational linguistics, automated reasoning, first-order representations, models.

1 What is Computational Semantics?

Computational semantics is a relatively new discipline that combines insights from formal semantics, computational linguistics, and automated reasoning. The aim of computational semantics is to find techniques for automatically constructing semantic representations for expressions of human language, representations that can be used to perform inference. In this paper we introduce computational semantics from a logic-oriented perspective.

We begin in Section 2 by discussing the most basic issue of all: what kinds of semantic representation are suitable for capturing the meaning of human language? Actually, there is no unique answer: it depends on what you want to do, on the level of detail at which you want to work, and on the linguistic phenomena you wish to analyse. Nonetheless, choices need to be made, and we shall argue that first-order logic is a sensible starting point.

One we've settled on a semantic representation language, how do we automate the process of associating semantic representations with expressions of human language? Essentially by using a syntactic analysis to guide the process of assigning semantic representations, and in Section 3 we discuss the two methods dominant in computational semantics for doing this: one based on unification, the other based on the lambda calculus. However we also need to cope with the ambiguities inherent in human language. Without context, many human language expressions can be assigned several meanings. So we also discuss two phenomena that lead to ambiguity, and outline ways of coping with them.

Finally, once we have semantic representations at our disposal, how can we use them to automate the process of drawing inferences? Section 4 discusses the use of techniques from automated reasoning (such as theorem proving and model generation) to implement consistency and informativeness checks. We also introduce the idea of generating minimal models of the meaning of natural language expressions.

But before turning to the details, a more general question should be addressed: why bother with computational semantics at all? There are at least two reasons. Firstly, computational semantics is potentially useful in such applications as information retrieval, information extraction, dialogue systems, question answering, interpreting controlled languages, and so on.

Secondly, it is likely to prove of increasing scientific importance. In the 30 years since the work of Richard Montague (Montague 1974) formal semantics has made substantial contributions to our understanding of the way human language works. Arguably,

however, further progress in semantics will depend on getting to grips with the interactions between various phenomena, and on better understanding the role played by inference. Such issues are inherently complex, and the use of computational tools will not merely be helpful, it will be vital.

2 Semantic Representations

2.1 First-Order Representations

Traditional formal semantic analyses of human language typically presuppose formalisms with high-expressive power (for example, higher-order logic augmented with modalities) but in computational semantics some variant of first-order logic is generally preferred. This choice is sensible for at least two reasons. First, as we shall discuss in Section 4, first-order theorem provers (and to a lesser extent, first-order model builders) now offer levels of performance which make them genuinely useful for certain reasoning tasks. Second, as we will show in this section, first-order logic is able to deal (at least to a good approximation) with a wide range of interesting phenomena. In short, first-order logic offers an attractive compromise between the conflicting demands of expressivity and inferential effectiveness.

Let's swiftly review first-order logic. Every first-order language has a vocabulary, telling us which symbols are used and how. Suppose we have a vocabulary consisting of the constants SUNSET-BOULEVARD, MULHOLLAND-DRIVE, the one-place relations WOMAN, AFRAID, and the two-place relations POLICE-REPORT, LOCATION, and CROSS. Such symbols are often called the non-logical symbols of the language. 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 ($\exists$, $\forall$), and the brackets plus the comma to group together symbols. The variables and constants are the terms of the language. The formulas of language are defined as follows:

1. If R is a symbol of arity n, and $\tau_1,\cdots,\tau_n$ are terms, then $R(\tau_1,\cdots, \tau_n)$ is a formula.
2. If τ_1 and τ_2 are terms, then $\tau_1=\tau_2$ is a formula.
3. If ϕ and φ are formulas then so are $\neg\phi$, $(\phi\wedge\varphi)$, $(\phi\vee\varphi)$ and $(\phi\rightarrow\varphi)$.
4. If ϕ 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 syntax we shall use throughout this article (we drop brackets if this will not lead to confusion). Here is an example of an English statement and its first-order translation:

A woman crosses Sunset Boulevard.

$\exists x(\text{WOMAN}(x)\wedge\exists y(y=\text{SUNSET-BOULEVARD}\wedge\text{CROSS}(x,y)))$

2.2 Interpreting First-Order Representations

First-order formulas are interpreted in models (these can be seen as abstract realizations of situations) with the aid of variable assignment functions (these can be seen as supplying extra contextual information). What do models look like? In set-theoretic terms, a model M is an ordered pair $(D;F)$ consisting of a domain D and an interpreta-

tion function F specifying semantic values in D. Here's a simple example (it should be clear that this model is a situation in which the formula just given is true):

D={d1,d2,d3} F(WOMAN)={d1}
F(CROSS)={(d1,d2),(d3,d2)}
F(SUNSET-BOULEVARD)=d2.

The crucial link between descriptions (first-order formulas) and situations (models) is made precise in the satisfaction definition. Formally, the satisfaction definition specifies a three place relation between a model $M=(D;F)$, a formula ϕ, and a variable assignment g (a function which maps variables to elements of D). The satisfaction relation is defined as follows:

$M, g \models R(\tau_1,\ldots,\tau_n)$	*iff*	$(I_F^g(\tau_1),\ldots, I_F^g(\tau_n)) \in F(R)$,
$M, g \models \varnothing f$	*iff*	not $M, g \models f$,
$M, g \models f \wedge y$	*iff*	$M, g \models f$ and $M, g \models y$,
$M, g \models f \vee y$	*iff*	$M, g \models f$ or $M, g \models y$,
$M, g \models f \rightarrow y$	*iff*	not $M, g \models f$ or $M, g \models y$,
$M, g \models \exists x f$	*iff*	$M, g' \models f$, for some x-variant g' of g,
$M, g \models \forall x f$	*iff*	$M, g' \models f$, for all x-variants g' of g.

In the first clause, I_F^g is F (c) if the term τ is a constant c, and g(x) if τ 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 $y \neq x$. Intuitively, variant assignments allow us to `try out' new values for the variable bound by the quantifier (here x).

Once the satisfaction definition has been given, the way is open to defining some fundamental inferential concepts. For example, a set of first-order formulas ϕ 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, ϕ is consistent if it describes a realizable situation). And a set of first-order formulas ϕ is informative if and only if it is *not* satisfied in all models (that is, ϕ is informative if what it describes rules out some situations).

But we defer our discussion of inference till Section 4. We must first consider whether first-order logic offers us the kind of expressivity needed in computational semantics.

2.3 First-order semantic representations

It is sometimes argued that first-order logic is too restrictive to model the semantics of human language in an interesting way. Such claims don't withstand scrutiny. To be sure, well-known results such as the Compactness Theorem and the Löwenheim-Skolem theorems show that first-order logic has expressivity limitations -but the limitations they reveal (such as its inability to distinguish infinite cardinalities) are usually tangential to the central concerns of computational semantics. As we shall now see, the kind of expressivity first-order logic offers opens the way to quite fine-grained

analyses of semantic phenomena- if we are prepared to be flexible about the kinds of entities which inhabit our models.

Modalities

At first glance, intensional phenomena (such as constructions involving necessity and possibility, or knowledge and belief) seem to take us beyond the realm of first-order logic, and many formal semanticists use various kinds of modal logic to cover these aspects of human language[1]. Here's an example. Extend first-order logic with the formula prefix operators $\Box$ (to express necessity) and $\Diamond$ (to express possibilities).

Thus we can now say things like:

Maybe Mulholland Drive is where the accident was.

$\exists x(x=\text{MULHOLLAND-DRIVE} \wedge \exists y(\text{ACCIDENT}(y) \wedge \Diamond \text{LOCATION}(x,y)))$

There must be a police report of the accident.

$\exists x(\text{ACCIDENT}(x) \wedge \Box \exists y(\text{POLICE-REPORT}(y,x)))$

At first blush, such examples may seem beyond the reach of first-order logic. But they're not. In fact, Kripke's celebrated semantics for modal logic is interesting precisely because it explains these rather mysterious looking intensional operators in terms of ordinary (extensional) first-order quantification. And it's simple to exploit Kripke's insight in a first-order semantic representation language. Add a second sort of entity to our models (call them `possible worlds' or `situations'). Add an accessibility relation R across these worlds. Add a one-place predicate symbol ACTUAL-WORLD to pick out the actual world. Add an extra argument place to each relation on ordinary individuals to relativise its interpretation to a particular world. Then translate away modalities as follows:

$(\Box \phi,w)^{m2f} = \forall v(R(w,v) \rightarrow (\phi,v)^{m2f})$,

$(\Diamond \phi,w)^{m2f} = \exists v(R(w,v) \wedge (\phi,v)^{m2f})$.

For example, the modal representation of `There must be a police report of the accident' becomes

$\exists w(\text{ACTUAL-WORLD}(w) \wedge \exists x(\text{ACCIDENT}(w,x) \wedge \forall v(R(w,v) \rightarrow \exists y(\text{POLICE-REPORT}(V,Y,X)))$.

In short, by letting models be mathematical pictures of richer ontologies (in this case, an ontology containing possible worlds) we have moved from modal logic back to ordinary first-order logic.

Tense and Aspect

Various temporal phenomena in human language (such as tense and aspect) can be analyzed using the modal apparatus of Prior-style tense logic, or various modal logics of intervals (both approaches are discussed in (van Benthem 1991)), but we are not forced to follow either route. For a start, both Prior-style tense logic and interval logics can be translated into first-order logic (in essentially the same way used above for ordinary modalities) so the way is open for either point-based or interval-based first-order semantic analyses. But other options are possible.

---

[1] See (Gamut 1991) for a textbook level introduction.

For example, we could take a Davidsonian route and enrich our models with primitive events and relations over them. This would allow us to take a sentence such as

A woman crossed Sunset Boulevard.

and represent it as:

$\exists x(\text{WOMAN}(x) \wedge \exists e(\text{CROSS}(e) \wedge \text{AGENT}(e,x) \wedge \text{THEME}(e,\text{SUNSET-BOULEVARD}) \wedge \exists t(\text{TLOC}(e,t) \wedge e<s))$.

As before, if we are willing to countenance a richer ontology (either one containing points of time, or intervals, or events) the way lies open to first-order analyses of the semantics of temporal constructions.

Plurals

How can we deal with the semantics of plurals in first-order logic? Here's one way: enrich our models with plural entities, add a one place predicate group to pick out such entities, and use a two-place relation member to indicate that an ordinary individual belongs to a group entity. Then we can represent the sentence

Two well-dressed men are drinking coffee.

using the first-order formula:

$\exists u(\text{GROUP}(u) \wedge \text{TWO}(u) \wedge \forall x(\text{MEMBER}(x,u) \leftrightarrow (\text{MAN}(x) \wedge \text{WELL-DRESSED}(x) \wedge \text{DRINKS-COFFEE}(x))))$.

We can further constrain the interpretation of the symbol two by formulating a meaning postulate:

$\forall u(\text{TWO}(u) \leftrightarrow \exists x \exists y(\text{MEMBER}(x,u) \wedge \text{MEMBER}(y,u) \wedge x \neq y))$.

This axiom states that a plural entity has the property two if and only if it has at least two distinct members.

2.4 The methodology of first-order modeling

We have seen that first-order logic offers expressive power relevant to the semantics of human language. We have also seen that the key to realizing this power is to be flexible about the kinds of entities we include in our models. To put it another way, first-order approaches to the semantics of human language go hand-in-hand with rich ontologies. This may be unpalatable to some philosophers. Arguably, however, the most promising methodology for the semanticist is to try to get to grips with the unruly ontology that human language seems to presuppose; to use Emmon Bach's (Bach 1986) phrase, the semanticist should engage in "natural language metaphysics". Indeed, it is arguable that this project is an indispensable *prelude* to philosophical analysis, though we won't pursue the point here.

Are there limitations to this style of first-order modeling? Yes. When we introduce new entities we have to introduce constraints governing how they behave. For example, we might want to constrain the accessibility relation on possible worlds to be reflexive, or constrain the precedence relation on events to be transitive, or insist that groups must have at least two 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 modeling is only an approximation. For example, if we developed the first-order approach to plurals sketched above in more detail, we would eventually find that we needed constraints that first-order logic couldn't handle[2].

But approximations are not to be despised, and many are remarkably good. Perhaps the best known approximation is the quasi-reduction of higher-order logic to first-order logic by introducing extra entities into models and constraining them to act like higher-order functions[3]. Not all the required constraints can be written in a first-order way of course (if they could, there would be no distinction between first- and higher-order logic) but enough first-order postulates can given to yield an approximation to higher-order logic that in many respects is better behaved than standard higher-order logic.

All in all, viewed from the perspective of contemporary computational semantics, the expressivity offered by first-order logic seems a reasonable starting point for semantic modeling. Indeed, when computational semanticists express doubts about first-order logic, their doubts don't center on traditional issues of expressivity, but on its (lack of) *dynamic potential*.

Here's an example. It's not easy to deal with discourse anaphora in first-order logic. Consider the discourse

A woman crosses Sunset Boulevard. She is afraid.

Now, this clearly means the same as the following first-order formula:

$\exists x(\text{WOMAN}(x) \wedge \text{CROSS}(x, \text{SUNSET-BOULEVARD}) \wedge \text{AFRAID}(x))$.

So there is no expressivity problem: first-order logic captures the content of this discourse.

But there is a trickier question -how can we systematically construct such representations? If we use (either of) the approaches described in the following section we will probably end up with

$\exists x(\text{WOMAN}(x) \wedge \text{CROSS}(x, \text{SUNSET-BOULEVARD})) \wedge \text{AFRAID}(x)$.

This representation is incorrect -the final occurrence of variable x is not bound by the quantifier, and thus is not linked with the variable x in WOMAN(x).

Several approaches to such issues have been explored. In Dynamic Predicate Logic (DPL) (Groenendijk & Stokhof 1991) the first-order satisfaction definition is changed so that the two representations just given mean exactly the same thing. In Discourse Representation Theory (DRT) (Kamp & Reyle 1993), on the other hand, we would represent the discourse using the following Discourse Representation Structure (DRS):

$[\{x,y\}, \{\text{WOMAN}(x), y=\text{SUNSET-BOULEVARD}, \text{CROSS}(x,y), \text{AFRAID}(x)\}]$.

Here the occurrence of x in AFRAID is linked to the x in $\{x, y\}$, and hence to the other occurrences of x.

Now, for present purposes it's not important to know how DRT and DPL manage to get things right -but it is important to realise that neither formalism increases the

---

[2] For a detailed discussion of this example, see (Lønning, 1997).

[3] For a good discussion of the second-order case, see (Enderton 1972); for full higher-order logic see (Doets & van Benthem 1983).

expressive power at our disposal. Neither DPL nor DRT is more expressive than ordinary first-order logic. All three formalisms can be freely inter-translated. To put it another way, they are notational variants.

But it would be *highly* misguided to conclude from this that the choice between them is merely a matter of convenience. It's not. For some purposes (notably, dealing with anaphora and presuppositions in a principled way) DRS notation (essentially a `flat' form of first-order logic without explicit quantifiers) is pretty much essential.

This is an instance of a lesson that comes up time and time again in computational semantics: we need to be flexible about the form our representations take. For example (as we shall learn when we discuss scope ambiguities in the following section) we sometimes need to think about representations in an abstract `underspecified' way. So when we argue that first-order representations are useful in computational semantics, our claim should be interpreted in this spirit. We're not arguing for blind devotion to orthodox first-order syntax: in practice it may be useful to freely move between orthodox syntax and variants (such as DRSs) as the need arises.

3 Computing Semantic Representations

How do we automate the process of assigning semantic representations to sentences of human language? That is, once we have fixed on a representation formalism (first-order logic, for example) how do we write programs which take human language sentences as input and return semantic representations as output? We will compare two standard approaches -one making extensive use of unification, and one based on the lambda-calculus. Both approaches require a grammar describing the syntactic structure of the fragment of language of interest. We first consider the unification-based approach, probably the most popular method in contemporary computational semantics.

3.1 Unification-based approaches

To guide the process of constructing semantic representations for a fragment of English, a fully specified syntax for the fragment is required. We will assume a syntactic analysis based on a collection of syntactic categories, whose interrelationships are described in phrase structure rules, and whose contents are represented in lexical entries. The categories themselves are represented as feature structures (familiar from various linguistic formalisms, and sometimes referred to as signs (Pollard, 1994)). Variables used for unification are represented by numbers in boxes. Here are some sample lexical entries:

$$\begin{bmatrix} \textit{phon}: \text{Mulholland Drive} \\ \textit{syn}: \text{pn} \\ \textit{sem}: \begin{bmatrix} \textit{index}: \boxed{1} \\ \textit{content}: \boxed{1}=\text{MULHOLLAND-DRIVE} \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textit{phon}: \text{walks} \\ \textit{syn}: \text{iv} \\ \textit{sem}: \begin{bmatrix} \textit{index}: \boxed{1} \\ \textit{content}: \text{WALK}(\boxed{1}) \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textit{phon}: \text{a} \\ \textit{syn}: \text{det} \\ \textit{sem}: \begin{bmatrix} \textit{index}: \boxed{3} \\ \textit{restr}: \boxed{1} \\ \textit{scope}: \boxed{2} \\ \textit{content}: =\boxed{3}(\boxed{1} \wedge \boxed{2}) \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textit{phon}: \text{woman} \\ \textit{syn}: \text{noun} \\ \textit{sem}: \begin{bmatrix} \textit{index}: \boxed{1} \\ \textit{content}: \text{WOMAN}(\boxed{1}) \end{bmatrix} \end{bmatrix}$$

Note that complex structures are used to instantiate the feature *sem* (that is, the *semantic* feature). Most lexical entries have an *index* feature designating a variable that needs to be equated with some other piece of (still missing) information. Determiners (here `a') have, in addition, features *restr* and *scope* whose task is to ensure correct placement of the two key components of the representation, namely the *restriction* and the *nuclear scope* (in the sentence `a woman walks', for example, linguists would call `woman' the restriction of the determiner `a', and `walks' its nuclear scope).

The phrase structure rules of the grammar direct the correct unification of all features in the semantic part of a sign. The rules have the form LHS → RHS, where LHS (left-hand side) is a non-lexical category, and RHS (right-hand side) a non-empty ordered set of lexical or non-lexical categories. The rules state how LHS categories can be expanded into a sequence of RHS categories. Here are some sample grammar rules.

$$\begin{bmatrix} phon: \boxed{1}+\boxed{2} \\ syn: \text{np} \\ sem: \begin{bmatrix} scope: \boxed{3} \\ content: \boxed{4} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} phon: \boxed{1} \\ syn: \text{det} \\ sem: \begin{bmatrix} index: \boxed{5} \\ restr: \boxed{6} \\ scope: \boxed{3} \\ content: \boxed{4} \end{bmatrix} \end{bmatrix} \begin{bmatrix} phon: \boxed{2} \\ syn: \text{noun} \\ sem: \begin{bmatrix} index: \boxed{5} \\ content: \boxed{6} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} phon: \boxed{1}+\boxed{2} \\ syn: \text{s} \\ sem: \begin{bmatrix} content \ \boxed{4} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} phon: \boxed{1} \\ syn: \text{np} \\ sem: \begin{bmatrix} index: \boxed{5} \\ scope: \boxed{3} \\ content: \boxed{4} \end{bmatrix} \end{bmatrix} \begin{bmatrix} phon: \boxed{2} \\ syn: \text{vp} \\ sem: \begin{bmatrix} index: \boxed{5} \\ content: \boxed{3} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} phon: \boxed{1} \\ syn: \text{np} \\ sem: \begin{bmatrix} index: \boxed{4} \\ scope: \boxed{2} \\ content: \exists\boxed{4}(\boxed{3} \wedge \boxed{2}) \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} phon: \boxed{1} \\ syn: \text{pn} \\ sem: \begin{bmatrix} index: \boxed{4} \\ content: \boxed{2} \end{bmatrix} \end{bmatrix}$$

Note how variables are used to pass on semantic information from daughter to mother categories. You may find it instructive to work through the analysis of the sentence `A woman walks.' If you do so you will obtain the feature structure

$$\begin{bmatrix} phon: \text{a woman walks} \\ syn: \text{s} \\ sem: \begin{bmatrix} content: \exists\boxed{1}(\text{WOMAN}(\boxed{1}) \wedge \text{WALK}(\boxed{1})) \end{bmatrix} \end{bmatrix}$$

The value assigned to the *content* feature clearly amounts to $\exists x(\text{WOMAN}(x) \wedge \text{WALK}(x))$, as we would expect.

The unification-based approach can be applied to deal with a wide range of semantic phenomena, and is very efficient. Many grammar formalisms which make use of feature structures build semantic representations in more-or-less the manner just sketched (Nerbonne, 1992a). Head-driven Phrase Structure Grammar (HPSG) is a case in point (Pollard, 1994, Frank & Reyle 1995).

But despite its merits, the unification-based approach makes no principled distinction between variables used for unification and variables used in semantic representations. This gives rise to problems in cases where chunks of semantic representation

need to be copied: a well known case is coordination. Even such a simple sentence as `Harry and Neal stare at the remains of the two cars' forces the value of the *index* feature of `stare' to unify with the indexes of `Harry' and `Neal'. This fails when the values are represented by constants, and yields an incorrect representation when they are represented as variables, and so the correct representation cannot be constructed.

Grammar engineers generally find a way around the problems posed by coordination (and other grammatical phenomena where copying is involved) by ad-hoc techniques applied to the lexicon or grammar rules. But there is a more principled approach to such difficulties: make use of the machinery provided by the lambda calculus.

3.2 Lambda-based Approaches

Let's use the lambda calculus as `glue-language' to combine semantic representations systematically. This approach is attractive from a grammar engineering perspective: as it distinguishes between the variables used to drive semantic construction and the variables used in semantic representations, it bypasses the problems raised by copying constructions such as coordination, thus making it easier to add a semantic component to large-scale grammars.

We first need to add `glue' to our representations: we will use the λ-operator to abstract over missing information, and the @-operator to express functional application. More precisely, all first-order formulas will be regarded as lambda expressions. Moreover, if x is a variable, and $\mathcal{F}$ is a lambda expression, then $\lambda x \mathcal{F}$ is also a lambda expression. In this expression the variable x is bound; it is these lambda-bound variables that drive the semantic construction process. Finally, if $\mathcal{F}$ and $\mathcal{A}$ are lambda expressions, then so is $(\mathcal{F}@\mathcal{A})$. Linking two expressions with an @ is essentially an instruction that the two representations have to be combined in the manner described below (with $\mathcal{F}$ as the functor and $\mathcal{A}$ as the argument).

We continue our practice of dropping brackets if no confusion arises.

Secondly, we have to reorganize our lexicon. Typical entries will now look like this:

$$\begin{bmatrix} \textit{phon}: \text{Mulholland Drive} \\ \textit{syn}: \text{pn} \\ \textit{sem}: \lambda x.x{=}\textsc{Mulholland-Drive} \end{bmatrix} \qquad \begin{bmatrix} \textit{phon}: \text{woman} \\ \textit{syn}: \text{noun} \\ \textit{sem}: \lambda x.\textsc{woman}(x) \end{bmatrix}$$

$$\begin{bmatrix} \textit{phon}: \text{a} \\ \textit{syn}: \text{det} \\ \textit{sem}: \lambda p.\lambda q.\exists x(p@x \wedge q@x) \end{bmatrix} \quad \begin{bmatrix} \textit{phon}: \text{every} \\ \textit{syn}: \text{det} \\ \textit{sem}: \lambda p.\lambda q.\forall x(p@x \rightarrow q@x) \end{bmatrix} \quad \begin{bmatrix} \textit{phon}: \text{walks} \\ \textit{syn}: \text{iv} \\ \textit{sem}: \lambda x.\textsc{walk}(x) \end{bmatrix}$$

And thirdly, the grammar rules. We still make use of variable unification, but we won't use it to manipulate variables in the semantic representation (so we don't get any unwanted interactions, and coordination will pose no difficulties):

$$\begin{bmatrix} phon: \boxed{1}+\boxed{2} \\ syn: \text{np} \\ sem: (\boxed{3}@\boxed{4}) \end{bmatrix} \rightarrow \begin{bmatrix} phon: \boxed{1} \\ syn: \text{det} \\ sem: \boxed{3} \end{bmatrix} \begin{bmatrix} phon: \boxed{2} \\ syn: \text{noun} \\ sem: \boxed{4} \end{bmatrix}$$

$$\begin{bmatrix} phon: \boxed{1}+\boxed{2} \\ syn: \text{s} \\ sem: (\boxed{3}@\boxed{4}) \end{bmatrix} \rightarrow \begin{bmatrix} phon: \boxed{1} \\ syn: \text{np} \\ sem: \boxed{3} \end{bmatrix} \begin{bmatrix} phon: \boxed{2} \\ syn: \text{vp} \\ sem: \boxed{4} \end{bmatrix}$$

$$\begin{bmatrix} phon: \boxed{1} \\ syn: \text{np} \\ sem: \lambda q.\exists x(\boxed{2}@x \wedge q@x) \end{bmatrix} \rightarrow \begin{bmatrix} phon: \boxed{1} \\ syn: \text{pn} \\ sem: \boxed{2} \end{bmatrix}$$

Let's consider an example. Here's what the approach would yield for `a woman walks':

$$\begin{bmatrix} phon: \text{a woman walks} \\ syn: \text{s} \\ sem: ((\lambda p.\lambda q.\exists x(p@x \wedge q@x)@\lambda x.\text{WOMAN}(x))@\lambda x.\text{WALK}(x)) \end{bmatrix}$$

That is, as we combine the various syntactic units, a sequence of function applications (the @s) records how the semantic representations are to be combined.

Let's now carry out the semantic combination (and in so doing get rid of all those @s). We do this using an operation called β-conversion (also known as β-reduction or lambda-conversion). β-conversion is the process of resolving all applications (expressions formed by the @-operator) by substituting the argument (the right-hand side of the @-operator) for the lambda-bound variables in the functor (the left-hand side of the @-operator). In the example just given this induces the following reduction steps:

$$((\lambda p.\ \lambda q\exists x(p@x \wedge q@x)@\ \lambda x.\text{WOMAN}(x))@\ \lambda x.\text{WALK}(x))=$$

$$(\lambda q.\exists x(\lambda x.\text{WOMAN}(x)@x \wedge q@x)@\ \lambda x.\text{WALK}(x))=$$

$$(\lambda q.\exists x(\text{WOMAN}(x) \wedge q@x)@\ \lambda x.\text{WALK}(x))=$$

$$\exists x(\text{WOMAN}(x) \wedge\ \lambda x.\text{WALK}(x)@x)=$$

$$\exists x(\text{WOMAN}(x) \wedge \text{WALK}(x)).$$

The process of β-conversion can make use of a process called α-conversion (renaming of bound variables) to avoid accidental variable bindings. β-conversion can either be performed during syntactic processing or in a distinct post-processing phase. There are standard procedures available that implement both α-conversion and β-conversion; see (Blackburn & Bos 2000).

We now have two ways of computing semantic representations. Note that the choice of method is independent of our style of syntactic analysis (we used the same grammar framework to illustrate both the unification-based and the lambda-based approaches). Which approach is better? We favour the lambda-based approach: although it is less efficient (we need to eliminate @s to obtain the final representation), its more disciplined treatment of variables better meets the requirements of serious grammar engineering.

Our brief survey has not covered all contemporary semantic construction methods. 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 (once you've used a formula to draw a conclusion, you can't use it again) and this enables it to be used to implement meaning composition. Formulations following this paradigm exist for Lexical Functional Grammar (van Genabith, 1999) and HPSG (Asudeh & Crouch 2002).

3.3 Dealing with Ambiguities

Expressions of human language are often highly ambiguous. There are many types of ambiguity, and all are of concern to computational semantics. We shall confine our discussion to two types: scope ambiguities and referential (or anaphoric) ambiguities.

Scope ambiguities arise when there are two or more scope bearing operator (such as quantifiers, negation, or modal expressions) in an utterance. Here's a standard example: the first interpretation is that the women witnessed possibly different accidents, the second is that there was a specific accident witnessed by all women:

Every woman witnessed an accident.

$$\{\ \forall x(\text{WOMAN}(x)\rightarrow\exists y(\text{ACCIDENT}(y)\wedge\text{WITNESS}(x,y))),$$
$$\exists y(\text{ACCIDENT}(x)\wedge\forall x(\text{WOMAN}(x)\rightarrow\text{WITNESS}(x,y)))\}$$

Don't be misled by the simplicity of this example: only two distinct representations are possible for this sentence, but in general the number of readings explodes exponentially as the number of scope-bearing operators increases. Moreover, in this example the two possible representations are related (the second implies the first) but this is by no means always the case[4]. Finally, in many syntactic frameworks this sentence would have only one plausible syntactic analysis, hence the syntax-driven semantic construction methods discussed in the previous section would only be capable of building one of the two representations. All in all, scope ambiguity is a serious problem, one that needs to be addressed by computational semantics.

One of the earliest treatment of scope ambiguities in a semantic formalism was Robin Cooper's influential method of quantifier storage (Cooper, 1983). Cooper designed special semantic representations called stores, containing the core semantics (with indexed free variables), and a set of unscoped quantifiers (using the indexes to control the binding of the free variables in the core semantic representation). Quantifiers could optionally enter the store during the computation of the semantic representation. At any point in the derivation, but normally at the end of the construction process, the quantifiers could be retrieved from the store and applied to the core representation, yielding an ordinary logical form. The different order in which retrieval could be carried out gave rise to the different scope possibilities. Bill Keller subsequently improved Cooper's work by introducing nested stores to deal with quantification in complex noun phrases (Keller, 1988). Many implementations use the storage technique to deal with quantifier scope ambiguities.

In recent years, however, the use of stores has been largely superceded by an approach known as semantic underspecification. The key idea shared by these newer approaches is to represent the meaning(s) of an ambiguous human language expression

---

[4] See (Gabsdil & Striegnitz 1999) for some nice examples.

in a compact way by *talking about the syntactic structure of the semantic representation*. First-order formulas, for example, can be regarded as trees. An underspecification formalism for first-order logic would provide tools for stipulating how the various subtrees in first-order representations should be nested. Nerbonne and Reyle seem to have been the first to coin the term underspecification in connection with semantic representations (Nerbonne 1992b, 1992a, Reyle, 1992, Frank & Reyle 1992, Reyle, 1993)[5].

Let's look at an example. Here's Hole Semantics (Bos 1996) at work:

Every woman witnessed an accident.

$< \{h_0, h_1, h_2, l_1, l_2, l_3\},$
$\{l_1{:}\forall x(\text{WOMAN}(x)\rightarrow h_1), l_2{:}\exists y(\text{ACCIDENT}(y)\wedge h_2), l_3{:}\text{WITNESS}(x,y)\},$
$\{l_1\leq h_0, l_2\leq h_0, l_3\leq h_1, l_3\leq h_2\}>$

The underspecified representations in Hole Semantics are tuples consisting of a set of entities (holes and labels), a set of labelled representations, and a set of scoping constraints. Holes represent unassigned scope. Holes can be plugged with labels as long as none of the constraints are violated. Scoping constraints are interpreted as dominance relations on nodes of labelled semantic representations (that is, first-order formulas viewed as trees). For example, the plugging that assigns l_1 to h_0 l_2 to h_1, and l_3 to h_2 yields the reading where `every woman' out-scopes `an accident'. On the other hand, the plugging that assigns l_2 to h_0, l_1 to h_2, and l_3 to h_1 yields the reading where `an accident' outscopes `every woman'.

Although these underspecified representations are more complicated than the logical forms we have used hitherto, such underspecified representations can be built using the machinery discussed in the previous section. Thus the use of hole semantics requires little new machinery. Either the unification-based (Richter & Sailer 1997), the lambda-based approach (Blackburn & Bos 2000, Bos, 2001), or linear logic (van Genabith et al. 1999) can be adopted.

Referential (or anaphoric) ambiguities are another source of problems for computational semantics. Depending on the context and situation, pronouns, proper names, definite description and other presuppositional expressions often have more than one potential antecedent to refer to. How do we represent these context-sensitive expressions, and how (and when) do we resolve them? Here's a (sketch of) one contemporary answer to such questions. In a classic paper, van der Sandt proposed treating presuppositional expressions on a par with anaphoric expressions (Van der Sandt, 1992) and introduced an intermediate representation (an extension of DRS notation from Discourse Representation Theory (Kamp, 1981)) which explicitly displayed all anaphoric information in unresolved form. Consider the following example:

Dan hasn't touched his bacon.

---

[5] Further examples of the approach include Minimal Recursion Semantics (MRS) (Copestake et al. 1995, Egg & Lebeth 1995), Pinkal's Underspecified Semantic Description Language (Pinkal 1996a, 1996b), Muskens' Ambiguous Logical Forms (Muskens 1995), Constraint Language for Lambda Structures (CLLS) (Egg et al. 1998) and work of several others authors, including Poesio (Poesio, 1994, 1996), and Schiehlen (Schiehlen 1997).

$[\{x\},\{\text{DAN}(x)\}]\alpha[\emptyset,\{\neg[\{z\},\{\text{MALE}(z)]\}\alpha[\{y\},\{\text{BACON}(y), \text{OF}(y,z)\}]\alpha[\{\emptyset,\{\text{TOUCH}(x,y)\}]\}]$

All context-sensitive information is marked by the α-operator, where the left-hand side DRS is the presuppositional information, and the right-hand side is the assertional part.

Here we have the proper name `Dan', the definite noun phrase `his bacon', which is lexically decomposed as "the bacon of him", resulting in nested α-DRSs. It is left open whether `Dan' refers to a previously mentioned entity in the discourse (anaphoric resolution), or whether it introduces a completely new object (accommodation). Similarly, whether `his bacon' refers to the bacon of Dan, or to somebody else's, is left unspecified.

The resolution algorithm for such unresolved representations traverses the DRS and decides, on encountering an α-DRS, whether to bind the anaphoric part to some accessible discourse referents, or whether to accommodate it to some accessible portion of discourse structure. Accommodation is possible either globally (that is, in the main DRS) or locally (at subordinated levels of discourse). For instance, for the above example, local accommodation of `his bacon' (in the scope of negation) would result in an interpretation where Dan didn't have bacon for breakfast (as in `Dan hasn't touched his bacon -in fact, he didn't have any bacon.')

We can't go deeper into this example, but three general points should be made. First, we mentioned in Section 2 that contemporary computational semantics takes a highly abstract view of representation; the use of DRSs annotated with special α-markers to distinguish presuppositional information (and indeed, the hole semantics example given earlier) illustrate this trend. Second, despite appearances to the contrary, we are still engaged in essentially the same business we started with: glueing together (variants of) first-order representations. Finally, the ideas sketched in this section have one over-riding virtue: they make good computational sense. On first encounter, the formalisms discussed here may seem complex. But they were designed for robustly practical reasons: they make it possible to incorporate a serious semantic component into large-scale grammars, and to use the result computationally. To give one example, the DORIS system (Bos, 2001, Bos, 2003) combines all the ideas discussed so far, together with the inference techniques discussed in the following section[6].

4 Inference

Inference plays many roles in computational semantics. In disambigutation it is used to filter out interpretations that make no sense, or to rank the likelihood of different interpretations. In generation it is used to test candidate sentences for suitability in a given situation. Many forms of inference (for example, probabilistic inference) are relevant to computational semantics. Here we discuss logical inference, focussing on the use of theorem provers and model builders.

---

[6] You can experiment with DORIS at www.coli.uni-sb.de/~bos/doris/.

4.1 Theorem Proving

In Section 2 we gave model-theoretic definitions of consistency and informativeness. The branch of logic called proof theory has developed many ways of recasting these concepts as symbol manipulation tasks. In its simplest form, 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.

Let Φ be a (finite) set of first-order formulas. Suppose we want to know whether Φ is consistent. If we give $\neg\Phi$ (the negation of the conjunction of the formulas in Φ) to a theorem prover, and the theorem prover finds a proof for this input, then we know that Φ is *not* consistent. (As it proved $\neg\Phi$, this must be true in all models, hence Φ is false in all models, that is, inconsistent.) On the other hand, suppose we want to know whether Φ is informative. If we give Φ to a theorem prover, and the theorem prover finds a proof for this input, then we know that Φ is *not* informative (as Φ was proved, Φ is true in all models, hence uninformative). Summing up: theorem provers offer us a negative handle on the problem of determining consistency, and on the problem of determining informativeness.

As is well known, first-order logic is undecidable. This means that it is not possible to write 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 are extremely efficient in practice, reaching levels of performance unheard of a decade ago. Moreover, current resolution provers even cope with formulas containing the equality symbol = (until recently, inference involving equality was difficult to handle efficiently). As semantic representations for human language typically make heavy use of equality, this is an important development.

Computational semantic applications for theorem provers include 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), and question answering (Bos & Gabsdil 2000). Some examples of state-of-the-art theorem provers are Vampire, SPASS, Bliksem, Otter, and Gandalf.

4.2 Model Generation

As their name suggests, 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 Φ, then Φ must be consistent) and the informativeness problem (if one successfully builds a model for $\neg\Phi$, then Φ must be informative). So model building is complementary to theorem proving (which offers negative handles on both problems).

Model building is a relatively new branch of automated reasoning, and model builders haven't reached the performance levels of theorem provers. Nonetheless, they have improved in the last few years, and are now starting to be useful in linguistic applications. Examples of model builders are MACE, IGNCS, and Kimba.

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 & Gabsdil 2000). In fact the DORIS system (www.coli.uni-sb.de/~bos/doris/) calls on both theorem proving and model building services to carry out positive and negative consistency and informativeness checks in parallel.

4.3 Minimal Models

While the most obvious use for model builders is to use them in tandem with theorem provers to perform consistency and informativity checks, 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 `flat', they are a level of representation from which it is 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.

Let's spell this out in a little more detail. Given a portion of discourse *D* and a goal *G*, the procedure we propose runs as follows:

> *1. Construct a first-order representation* Φ *for D, taking into account background knowledge.*
>
> *2. Attempt to build a model for* Φ*:*
>
> > *(a) Give* Φ *to a model builder, possibly resulting in a model M (for consistent representations* Φ*).*
> >
> > *(b) Give* $\neg\Phi$ *to a theorem prover, possibly resulting in a proof (for inconsistent representations* Φ*). Fail.*
>
> *3. Use M to extract information required for G.*

Step *1* presupposes tools that construct semantic representations and carry out ambiguity resolution (in short, the sort of tools we have been discussing in this paper) together with parsers, speech recognisers and so forth; nowadays a wide range of such tools are freely available.

We remark that the semantic representations we build need not to be in conventional first-order syntax. For example, implementations of translations of DRSs to conventional first-order syntax are available (Bos, 2001). Step 1 also presupposes that a certain amount of background knowledge has been formalised in (some variant of) first-order logic.

Step *2* attempts to construct a model for Φ. Ideally, it should be realized by running *2(a)* and *2(b)* in parallel. Theorem provers weed out inconsistent representations Φ (the result of ambiguity resolution as part of the previous step of processing might result in several alternative representations, some of them inconsistent), and the model

builders find minimal models for **Φ**. The concept of minimality is of importance here; it ensures that no redundant information is incorporated in *M*.

Step *3*, finally, links models to actions. That is, it uses the information provided by the models to carry out certain tasks, such as database retrieval or the performance of physical actions. Once you know what you're looking for in the model, this is the easiest part of the procedure. For more sophisticated queries, one can also use a model checker (Blackburn & Bos 2000).

To illustrate why this is useful, consider the sentence `The Boston office called.' There are two semantical problems with this example. Offices do not `call', but of course a human being on hearing this sentence unconsciously coerces `the Boston office' to `an employee of the Boston office' or `someone in the Boston office'. Second, the relation between `Boston' and `office' is not specified in the compound nominal (depending on the background information, it might mean either `office located in Boston' or `office that handles Boston-related business').

Now, instead of trying to build some first-order representation, let's apply the above algorithm to generate a model that gives us a small `picture' of what seems to be going on.

Let's suppose we have the following background information at our disposal:

$$\{\ \text{BOSTON}(b),\ \text{OFFICE}(o),\ \text{IN}(o,b),\ \text{PERSON}(p),\ \text{EMPLOY}(o,p),$$
$$\forall x\forall y(\text{IN}(x,y)\rightarrow\text{NN}(y,x)),\ \forall x\forall y(\text{EMPLOY}(x,y)\rightarrow\text{REL}(y,x))\}.$$

Then a minimal model for the knowledge base and the sentence would be:

```
D={b,o,p} F(boston)={b}
          F(office)={o}
          F(person)={p}
          F(in)={(o,b)}
          F(employ)={(o,p)}
          F(rel)={(p,o)}
          F(nn)={(b,o)}
          F(call)={p}.
```

In a sense, we have used the model builder to `guess' what the world must be like (given certain background information) for the sentence to be true[7].

The idea of using model builders to `guess' pictures of the world has affinities with a special kind of inference known as abduction. Abduction can be thought of as using deductive rules backwards to provide explanations; for example, an abductive inference would use the information $\{p\rightarrow q,\ q\}$ to hypothesize p as a plausible explanation for q (for q follows from $p\rightarrow q$ by modus ponens given p). In the `Boston office' example, an abduction based approach would use the background information to build a plausible first-order representation (Hobbs et al. 1990). For example, it might propose this:

$$\exists x(\text{PERSON}(x)\wedge\text{CALL}(x)\wedge\exists y(\text{REL}(x,y)\wedge\text{OFFICE}(y)\wedge\exists z(\text{BOSTON}(z)\wedge\text{NN}(z,y)))).$$

---

[7] Similar uses of model builders for discourse interpretation may be found in (Gardent & Webber 2001, Richter & Sailer 2000, Gardent & Konrad 2000).

The ideas underlying abduction and model generation algorithm are clearly rather similar.

Their computational properties are slightly different though. In model generation, alternatives are available until they become inconsistent. In weighted abduction, once a proof is found, its result is stored in the knowledge base and reconsideration of the proof is not possible anymore.

5 Conclusion

In this paper we have given an overview of computational semantics. Our account has given a prominent role to logic, especially first-order logic: we've assumed that the business of the computational semanticist is to build up detailed meaning representations for sentences, that these meanings will be represented in (some version of) first-order logic, and that various forms of first-order inference will play a role in semantic processing. We should emphasise, however, that other approaches are both possible and interesting. For example, shallow processing and probabilistic inference may be faster for many applications.

What of the future? Many strands of research (such as developments in lexical semantics) are relevant to computational semantics, but in keeping with our logic-oriented perspective we'll confine our discussion to likely developments in the area of inference.

One emerging theme is the use of description logics. Description logics are restricted fragments of first-order logic; they are typically decidable and some excellent implementations exist. While they lack the expressive power to deal with natural language semantics in full generality, for some applications they suffice[8]. In such cases, description logic based inference can be extremely efficient, and there will probably be further experiments with their use in the near future.

Another theme is direct reasoning with underspecification formalisms. Is it possible to perform useful inference efficiently on such formalisms without expanding them out?[9]

BIBLIOGRAPHY

Asudeh, A., Crouch, R. (2002) "Glue Semantics for HPSG" in F. van Eynde, L. Hellan, D. Beermann (eds), *Proceedings of the 8th. International HPSG Conference*. Stanford: CSLI Publications.

Bach, E. (1986) "Natural language metaphysics" in. R.B. Marcus, G.J.W. Dorn, P. Weingartner (eds), *Logic, Methodology and Philosophy of Science VII*, North Holland, 573-595.

Blackburn, P., Bos, J. (2000) "Representation and Inference for Natural Language. A First Course in Computational Semantics". Draft available at http://www.comsem.org

Blackburn, P., Bos, J., Kohlhase, M., de Nivelle, H. (2001). "Inference and Computational Semantics" in H. Bunt, R. Muskens, E: Thijsse (eds), *Computing Meaning*, vol. 2, Kluwer, 11-28.

Bos, J., (1996) "Predicate Logic Unplugged" in P. Dekker, M. Stokhof, (eds), *Proceedings of the Tenth Amsterdam Colloquium*, Amsterdam: ILLC/Dept. of Philosophy, University of Amsterdam, 133-143.

---

[8] See, for example, (Ludwig et al. 2000).

[9] For preliminary work in this direction see (Reyle 1993, Reyle 1995, König 1997, Muskens 1997, Jaspars and Van Eijck 1998).

Bos, J., (2001) "DORIS 2001: Underspecification, Resolution and Inference for Discourse Representation Structures". in P. Blackburn, M. Kohlhase (eds) *ICoS-3, Inference in Computational Semantics*, 117-124.

Bos, J., (2003). "Implementing the binding and accommodation theory for anaphora resolution and presupposition projection", *Computational Linguistics*, (in press).

Bos, J., Gabsdil, M. (2000) "First-order inference and the interpretation of questions and answers" in M. Poesio, D. Traum (eds), *Proceedings of Götalog 2000, Fourth Workshop on the Semantics and Pragmatics of Dialogue*, 43-50.

Cooper, R. (1983). *Quantification and Syntactic Theory*. Dordrecht: Reidel.

Copestake, A., Flickinger, D., Malouf, R., Riehemann, S., Sag, R. (1995). "Translation using Minimal Recursion Semantics" in *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*, Leuven: University of Leuven, Belgium.

Doets, K., van Benthem, J. (1983). "Higher-order logic" in D.M. Gabbay, F. Guenthner, (eds) *Handbook of Philosophical Logic*, vol. 1, Dordrecht: Reidel, 275-329.

Egg, M., Lebeth, K. (1995). "Semantic underspecification and modifier attachment ambiguities" in J. Kilbury, R. Wiese (eds) *Integrative Ansätze in der Computerlinguistik*, Düsseldorf: Seminar f¨ur Allgemeine Sprachwissenschaft, 19-24.

Egg, M., Niehren, J., Ruhrberg, P., Xu, F. (1998). "Constraints over Lambda- Structures in Semantic Underspecification" in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*. Montreal: Universite de Montreal, 353-359.

Enderton, H.B. (1972). *A Mathematical Introduction to Logic*. New York: Academic Press.

Frank, A., Reyle, U. (1992). "How to Cope with Scrambling and Scope" in Görz, G. (ed), *Konvens 92, Informatik aktuell*. Berlin: Springer, 178-187.

Frank, A., Reyle, U. (1995). "Principle Based Semantics for HPSG" in *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics*, Dublin, 9-16.

Gabsdil, M., Striegnitz, K. (1999). "Classifying Scope Ambiguities" in C. Monz, M. de Rijke (eds) *ICoS-1, Inference in Computational Semantics*, Amsterdam: Institute for Logic, Language and Computation (ILLC), 125-131.

Gamut, L.T.F. (1991) *Logic, Language, and Meaning. Volume II. Intensional Logic and Logical Grammar*. The University of Chicago Press.

Gardent, C., Konrad, K. (2000). "Interpreting definites using model generation". *Journal of Language and Computation*, 1(2), 193-209.

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

Groenendijk, J., Stokhof, M. (1991). "Dynamic Predicate Logic". *Linguistics and Philosophy*, 14, 39-100.

Hobbs, J. R., Stickel, M. E., Appelt, D., Martin, P. (1990). "Interpretation as abduction". *Technical Report* 499.

Jaspars, J., Van Eijck, J. (1998). *Ambiguity and Reasoning* (unpublished)

Kamp, H. (1981). "A Theory of Truth and Semantic Representation". *Formal Methods in the Study of Language*, 277-322.

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

Keller, W.R. (1988) "Nested Cooper Storage: The Proper Treatment of Quantification in Ordinary Noun Phrases" in. U. Reyle, C. Rohrer (eds), *Natural Language Parsing and Linguistic Theories*. Dordrecht: Reidel.

König, E., Reyle, U. (1997). "A General Reasoning Scheme for Underspecified Representations" in H.J. Ohlbach, U. Reyle (eds), *Logic and its Applications. Festschrift for Dov Gabbay. Part I*. Dordrecht: Kluwer.

Lønning, J.T. (1997). "Plurals and collectivity" in J. van Benthem, A. ter Meulen (eds), *Handbook of Logic and Language*. Amsterdam: Elsevier, 1009-1053.

Ludwig, B., Görz, G., Niemann, H. (2000). "An inference-based approach to the interpretation of discourse". *Journal of Language and Computation*, 1(2):261-276.

Montague, R. (1974). *Formal Philosophy: Selected Papers of Richard Montague* (edited and with an introduction by R. Thomason) Yale University Press.

Muskens, R. (1997). "Underspecified representations". *Technical Report 95*, CLAUS.

Muskens, R. A. (1995). "Order-independence and Underspecification" in Groenendijk, J. (ed), *Ellipsis, Underspecification, Events and More in Dynamic Semantics*, 17-34. Dyana Deliverable R2.2.C.

Nerbonne, J. (1992). "A Feature-Based Syntax/Semantics Interface". *Research Report* RR-92- 42, Saarbrücken: DFKI.

Nerbonne, J. (1992). "Constraint-Based Semantics" in M. Stokhof, P. Dekker (eds), *Proceedings of the Eighth Amsterdam Colloquium*, ILLC, University of Amsterdam, 425-443.

Pinkal, M. (1996). "Radical Underspecification" in M. Stokhof, P. Dekker (eds), *Proceedings of the Tenth Amsterdam Colloquium*, , ILLC/Dept. of Philosophy, University of Amsterdam, 587-606

Pinkal, M. (1996). "Wie die Semantik arbeitet. Ein unterspezifiziertes Modell" in G. Harras, M. Bierwisch (eds), *Wenn die Semantik arbeitet*, Tübingen: Max Niemeyer Verlag, 57-88.

Poesio, M. (1994). "Ambiguity, Underspecification and Discourse Interpretation" in H. Bunt, R. Muskens, G. Rentier (eds*), International Workshop on Computational Semantics*, University of Tilburg, 151-160.

Poesio, M. (1994). "Disambiguation as (Defeasible) Reasoning about Underspecified Representations" in P. Dekker, M. Stokhof (eds), *Proceedings of the Tenth Amsterdam Colloquium*, ILLC/Dept. of Philosophy, University of Amsterdam, 607-623.

Pollard, C., Sag, I.A. (1994). *Head-Driven Phrase Structure Grammar. Studies in Contemporary Linguistics*. The University of Chicago Press.

Ramsay, A., Seville, H (2000). "Models and discourse models". *Language and Computation*, 1(2), 167-181.

Reyle, U. (1992). "Dealing with Ambiguities by Underspecification: A First Order Calculus for Unscoped Representations" in M. Stokhof, P. Dekker (eds), *Proceedings of the Eighth Amsterdam Colloquium*, ILLC/Dept. of Philosophy, University of Amsterdam, 493-512.

Reyle, U. (1993). "Dealing with Ambiguities by Underspecification: Construction, Representation and Deduction". *Journal of Semantics*, 10:123-179.

Reyle, U. (1995). "On Reasoning with Ambiguities" in *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics*, Dublin, 1-8.

Richter, F., Sailer, M. (1997). "Underspecified semantics in HPSG" in H. Bunt, L. Kievit, R. Muskens, M. Verlinden (eds), *Proceedings of the Second International Workshop on Computational Semantics*, Tilburg, 234-246.

Schiehlen, M. (1997). "Disambiguation of Underspecified Discourse Representation Structures und Anaphoric Constraints" in *Second International Workshop on Computational Semantics*, University of Tilburg.

van Benthem, J. (1991) *The Logic of Time*. Kluwer (second edition).

Van der Sandt, R.A. (1992). "Presupposition Projection as Anaphora Resolution". *Journal of Semantics*, 9, 333-377.

van Genabith, J., Frank, A., Crouch, R. (1999). "Glue, underspecification translation" in H. C. Bunt, E. G. C. Thijse (eds), *IWCS-3, Third International Workshop for Computational Semantics*, 265-279.

Patrick Blackburn is Directeur de Recherche (Director of Research) at INRIA, France's national organisation for research in computer science. He is based at LORIA, a large research institute in Nancy, and is currently attached to the Langue et Dialogue (Language and Dialogue) team. His research centers on logic and its applications in cognitive and computer science, particularly applications that have something to do with ordinary human languages.

Address: INRIA Lorraine. 615, rue du Jardin Botanique, 54602 Villers lès Nancy Cedex, France. E-mail: patrick@aplog.org

Johan Bos is a researcher at the Language Technology Group of the University of Edinburgh, Scotland. Trained as computational linguist (he received his PhD entitled "Underspecification and Resolution in Discourse Semantics" from the University of the Saarland, in 2001), his main focus of research is the area of computational semantics, with particular interests in the speech-semantics interface, aspects of knowledge representation and inference, dialogue phenomena, and discourse analysis. Currently, he is working on the design of intelligent human-machine dialogue systems, by combining linguistic and logical methods with techniques borrowed from automated deduction.

Address: School of Informatics, The University of Edinburgh, 80 South Bridge, Edinburgh EH8 9LW.
E-mail: jbos@inf.ed.ac.uk