Chapter 4 Computing with Meaning

Maarten de Rijke

Abstract Let's go back to Van Benthem's chapter. There, we saw many examples of everyday scenarios of communication in action, ranging from very simple ones (such as asking for directions) to far more intricate ones where the flow of information is far from immediately obvious (such as cryptographic protocols). Understanding these scenarios from a logical point of view requires increasingly more sophisticated representation formalisms, in which the information states of the parties involved get richer and in which the reasoning performed gets more complex. From a computational point of view there are many challenging aspects to these scenarios, having to do with computational complexity, with algorithms for reasoning, with algorithms for generating representations of natural language expressions, and with the interaction between the logical and linguistic processes that seem to be underlying them. Indeed, these are the issues we face whenever we want to communicate with language in a computational setting, whether in a multi-agent setting or in a setting with a single agent who is accessing natural language texts.

1 Introduction

Reasoning is a notoriously hard problem, and so is the process of building representations of natural language texts that can usefully be fed to our reasoning engines. As we know from modern logic since the 1930s in the fundamental work of Gödel and Turing, making a logical language reasonably expressive can make valid reasoning in it *undecidable*: i.e., beyond the scope of any algorithm whatsoever. So, there is no miracle cure. We rather have to understand the *balance* between expressive power and computational complexity, looking for logical architectures that represent some optimum.

How far can we go? At which point does reasoning become prohibitively hard? And what level of analysis of a piece of text is attainable given certain real-world constraints on our computational resources? Rich representation structures are hard to generate and hard to manipulate;

systems that need to work in real-time can only handle relatively shallow information structures. Indeed, there is an inevitable tension between informational expressiveness and computational complexity. And this tension shows in all current natural language systems. The richer your syntactic and semantic analyses, the more complex the workings of any corresponding system for handling meaning or communication. Therefore, we see many kinds of *shallow*, *partial*, and *underspecified* representations: one tries to do tasks at the shallowest level possible.

This, then, is our main interest: what is the balance between inference and representation? There are many aspects to this: When does it matter to go for deep levels of analysis and rich representations? Does the quality of results for a given task improve substantially when we go for much richer representations?... These are very abstract questions, so let's make things more concrete.

An Example: Entailment Checking

We take a brief look at the following task: determining entailment relations between natural language texts. This is a task that naturally occurs as a subtask in a number of modern natural language processing systems. In dialogue systems [37], for instance, we want the system to be *informative*, that is, the system should only assert a piece of information if it does not follow from the information that has already been exchanged with the user. Another example is provided by multi-document summarization [9, 44], where one may try to build a summary by starting from a given document and only adding parts from other documents that are not implied by the information that we have already included. Figure 1 shows an example of two segmentized documents, both covering a plane crash in Taiwan. A superficial glance reveals some obvious entailments; for instance, **AP 2** is at least as informative as **Reuters 4** and as **Reuters 5**. On the other hand, it seems clear that **AP 2** does not contain the specific information mentioned in **Reuters 3**.

Now, what kind of entailment checking is appropriate? And what kind of algorithms should we use? And what kind of representations should they work on? In deciding these matters, the following criteria are among the main ones:

- the robustness and coverage of methods for generating representations that our system can work on,
- the computational costs (and behavior) of the entailment checks, and

Figure 1: Two Segmentized Documents (Topic 6).

Reuters (31 Oct 2000 16:39 GMT)	AP (31 Oct 2000 16:25 GMT)
Reuters 1: TAIPEI (Reuters) - A Singapore Airlines plane bound for Los Angeles crashed during a typhoon at Taiwan's international airport on Tuesday, an airport police official said. **Reuters 2:** It was not immediately known how many of the 159 passengers and 20 crew were killed or injured, Civil Aeronautics Administration deputy director Chang Kuo-cheng told reporters. "The plane burst into flames and exploded shortly after takeoff," an airport police official told reporters. **Reuters 3:** Local television was reporting that over 120 injured had been taken to hospital. **Reuters 4:** The SIA Boeing 747-400 was taking off during a storm and hit by strong winds. It hit two other planes on the tarmac, including a China Airlines plane, police said. A Taiwan vice transport minister said no one was on board the other two planes. **Reuters 5:** The injured were rushed to hospital. No other details were immediately available. ⋮	**AP 1:** TAIPEI, Taiwan (AP) - A Singapore Airlines jetliner bound for Los Angeles crashed on takeoff in a storm Tuesday night and slammed into another plane on the runway, a Taiwanese official said. **AP 2:** There were 179 people on board Singapore Airlines Flight SQ006, which local media reports said was a 747. It was not immediately known how many people were hurt or killed, but local media reports said some injured people were being taken to the hospital. Strong winds seemed to have forced the plane down. There was an explosion as it struck a China Airlines plane on the runway at Taipei's Chiang Kai-shek International Airport, emergency official Wu Bi-chang said. Local media reports said the China Airlines plane was empty. **AP 3:** The crash occurred at 11:18 p.m. local time, and rescue workers were being dispatched to the scene, Wu said. Minutes later, the flashing lights of rescue vehicles were visible on the wet tarmac. Local media reports said there was a fire on the runway after the crash but that it had been extinguished. ⋮

- the type of outcomes or output that we want to have.

Below we discuss these criteria in a bit more detail.

Generating Representations. Whichever tool we use for entailment checking, it needs to operate on *representations* of the input documents. Semanticists have tended to opt for rich representation formalisms that are able to capture as many relevant aspects as possible; Dekker's and Van Eijck's chapters are examples of this. In practice, 'rich' often means 'includes (at least) first-order logic.' What does it take to generate first-order logic representations of documents? Traditionally, this involves a number of levels, including syntactic structure, logical form, and some form of contextual interpretation [2]. Despite important advances, relevant tools (such as parsers) lack the robustness and coverage needed to efficiently generate deep semantic representations of arbitrary natural language documents. Moreover, the traditional demand that representations be precise and unambiguous may lead to representations whose size is exponential (or worse) in the size of the input document. Practicable methods for generating first-order representations are rare.

An obvious way out is to turn to more light-weight representations that can be obtained by more shallow and more robust language processing techniques. Partial parsing or chunk parsing can be used to build such representations in an efficient and robust manner, while avoiding full disambiguation [1, 33].

Bags of (stemmed) words, possibly filtered through a stopword list, are even more shallow representations of natural language documents. At the cost of giving up virtually all syntactic structure, bag-of-words representations provide very concise representations that are easy to generate, and that are typically used for large text collections [8].

Computational Costs. How hard is it to reason with pieces of information in a given representation format? If first-order logic is the representation formalism of choice, the entailment problem is obviously undecidable. Admittedly, recent advances in first-order theorem provers and model generators do seem to make them of practical use in some classes of linguistic problems [11]. And there is hope that the development of test suites for, e.g., discourse understanding will allow the tuning of automated inference systems to excel at strategies that support efficient inference for semantics [24]. But, we are dealing with an undecidable problem; in practice this means that minor variations in do-able instances may cause time outs and exploding memory usage.

There are two obvious replies to this problem. One is to ignore it and to accept the fact that there are problem instances for which the available computational resources are guaranteed not to suffice. The other is to stick to wide coverage but to simplify the reasoning task so that it is guaranteed to behave well on each problem instance.

Type of Outcomes. There is another fundamental issue here. Most reasoning methods in computational semantics are based on strict, binary logical reasoning, and if they produce an outcome at all, it will be a strict yes or no. As any textbook on artificial intelligence will explain, there are many reasons why approximate reasoning should be preferred in some cases. For the purposes of entailment checking in computational semantics, two reasons are particularly relevant.

First, in many practical situations, such as document summarization, we are content with *approximate* answers, and prefer approximate answers to having no answer at all. Second, as we will see below (Section 3), usually, the strict binary entailment relation only holds between text segments $s_{i,d}$ in document d and $s_{j,d'}$ in document d' (in that order) whenever $s_{i,d}$ is a copy of $s_{j,d'}$ or an extension of a copy of $s_{j,d'}$. In other words, if only Boolean answers are allowed, the entailment relation may be too sparse (i.e., hold between too few pairs of text segments) to be of any practical use.

Which entailment checking method works best? We offer no universal answer, but in Section 3 we report on the development of a test set aimed at evaluating such methods.

Motivation

Why are we interested in the balance between inference and representation in informational tasks? Shouldn't computer scientists be spending their time devising information accessing methods that are better, faster, and more powerful? They should, and many of us actually do. But many researchers want to understand the inherent strengths and weaknesses of their methods. The motivation for this quest is three-fold. First of all, there is *intellectual curiosity.* From the early Greeks' experiments in working out the circumference of the earth, to the latest probings beyond the borders of the observable universe, man has asked questions about our universe, and our place in it [20]. Similarly, when dealing with today's core asset — information —, computer scientists want to discover what can be inferred from it and what cannot, and how costly it is when it can.

A second reason for being interested in the balance between inference and representation is to *encourage new paradigms.* Knowledge representation and automated reasoning systems have now reached a level of sophistication where they can be put to good use, as witnessed, for instance, by the DORIS system developed by Bos [12]. Understanding just when and why such tools can be used for informational tasks, would be important for researchers from the automated reasoning and computational logic communities. Understanding the limitations of existing representation and inference methods will help us to develop alternatives.

A third reason is simply to *avoid futile efforts.* The (short) histories of computational logic and natural language processing are full of ideas and methods that look nice in principle, but don't work. Indeed, many researchers try to use deep analyses where these simply don't work. The more we understand these matters, the less we will waste our time and energy on such endeavors.

So much for our key interest and its motivations. Let's turn now to the type of answers that we'd like to see and how we go about finding them.

Finding Answers

Since our main interest lies with inference and representation, and the balance between them, the sort of results that we're after concern methods for generating representations and for reasoning with them, as well as means for *evaluating* these methods. But how do we evaluate? Theoretical studies are important to us because they can provide us with an analysis of the computational complexity and termination behavior of our reasoning methods. They can provide us with an analysis of the expressive power of our representation languages. Theoretical analyses can also give us insights into the descriptive requirements of the domains we're trying to model [53], or in the mathematical relationships between, for instance, different parsing methods. For further examples of theoretical studies that are relevant for our purposes, see the chapters by Van Benthem and by Venema.

While theoretical studies paint part of the picture, for a fuller understanding of the interaction between inference and representation, empirical studies are needed as well. Theoretical tools are not always available to answer the questions we want answered. Theoretical analyses can produce results on worst-case computational complexity of reasoning methods, but this may be too coarse and tends to focus exclusively on worst-case scenarios. Moreover, actual systems that implement algorithms will have the same algorithmic complexity, which is not very useful in determining the effectiveness of, for instance, optimizations.

Moreover, some questions are inherently experimental and cannot be addressed by theoretical tools. For instance, how far can we push first-order inference techniques in natural language understanding? Empirical testing provides insights not produced by studying theoretical analyses: it directly gives resource consumption, in terms of computation time and memory use. It factors in all the pieces of a system, not just the basic algorithms itself. Moreover, empirical testing can be used not only to compare different systems, but also to tune a system or to show what sort of inputs the system handles well, or poorly.

As we will see below, several kinds of inputs can be used for empirical testing, ranging from randomly generated problem sets to standard document collections equipped with a 'gold standard' of the correct answers. What we actually want to *measure* while carrying out experimental analyses, depends on many ingredients, including the specific task that an algorithm or system is supposed to address. To evaluate an inference engine, for instance, the most common measures of performance are time and space: the shorter the time it takes the system to yield a proof (or a counterexample), the smaller the space used, the better the system is considered to be. In other settings, other metrics besides time and space are also of interest. For a system designed for building syntactic or semantic representations of natural language texts, one measures how often the system gets it right, and how many of the right answers it produces. In a system designed for providing information retrieval we are dealing with a user's information request that can be vague; the retrieved documents may have to be ranked according to their relevance to the query. Thus, information retrieval systems require the evaluation of how precise the answer set is.

This Chapter

We have indicated how simple communication scenarios give rise to a wide range of computationally interesting issues, even in the setting of a single agent that deals with natural language texts. Our focus is on the balance between inference and representation, and our goal here is to make this theme more concrete by reviewing some relevant research efforts within the *Logic in Action* project and its follow-ups. While we will not ignore theoretical analyses, our emphasis will be on experimental evaluation. We start by looking at reasoning; after that we consider reasoning with actual representations of natural language texts, and, finally, we look at experimental evaluations by means of real-world tasks. One word of warning: much of the material below is concerned with ongoing work, where the final (and in some cases: even the first) answers have not been found yet.

2 Reasoning

The working logical representation language of any logic introduction is *first-order predicate logic*, a reasonably expressive representation formalism for natural language. While it has been argued that it is too *weak* to be used as a universal representation formalism for natural language, there is a clear sense in which it is too strong: no effective algorithm can test its valid forms of reasoning in their entirety. There are logical systems that put the balance differently. A central example is *modal logic*, which restricts the power of first-order quantification to ‘locally accessible’ objects in the domain of discourse, thereby achieving a consequence problem which is decidable. Let’s have a look at some theoretical and experimental aspects of modal logic as they relate to our special interests.

Theoretical Aspects: Expressive Power

Recall from Van Benthem’s chapter that formulas of the basic propositional modal language are built up from atomic proposition letters, the boolean connectives $\neg$ and $\wedge$, and the modal operators $\Diamond$ and $\Box$. Models for this language are triples of the form (W, R, V), where W is the domain of discourse, R is a binary relation on W (often called the accessibility relation), and V is a valuation function that indicates how atomic information is distributed over the objects in the domain of discourse. Truth of a modal formula ϕ at a state or object w (in symbols: $M, w \models \phi$) is defined by recursion, with the following key clauses:

- $(W, R, V), w \models \Diamond\phi$ if there exists $v \in W$ with Rwv: $(W, R, V), v \models \phi$;
- $(W, R, V), w \models \Box\phi$ if for all $v \in W$ with Rwv: $(W, R, V), v \models \phi$.

Here, then, is the essential definition; see also Venema’s chapter.

Definition 1 A *bisimulation* between two models $M_1 = (W_1,\ R_1,\ V_1)$ and $M_2 = (W_2,\ R_2,\ V_2)$ is a non-empty relation Z between the domains W_1 and W_2 of M_1 and M_2, respectively, that satisfies the following conditions:

1. Z-related objects satisfy exactly the same proposition letters;
2. if w_1 and w_2 are Z-related and a transition can be made from w_1 to v_1 via the accessibility relation in M_1, then, in M_2, a similar move can be made along the accessibility relation from w_2 to some state v_2 in M_2 that is Z-related to v_1;
3. similar to the previous item, but now every move along the accessibility relation in M_2 should mimicked by a similar move in M_1.

Let us call two objects in two models *bisimilar* if there exists a bisimulation between them. The expressive power of modal formulas can now be captured as follows. First, we note that modal formulas can be mapped into first-order predicate logic, in a satisfiability preserving way, simply by observing that proposition letters are nothing but unary predicates and that the above truth definitions for $\Diamond$ and $\Box$ can be transcribed in first-order logic. Then the following equivalence may be proved.

Theorem 2 *Let $\phi(x)$ be a first-order definable property of states in modal models (W, R, V). Then, $\phi(x)$ is equivalent to a modal formula if and only if $\phi(x)$ cannot distinguish between bisimilar objects.*

In other words, the modally definable properties are exactly those that cannot distinguish between bisimilar states. The beauty of this result is not just that it gives us an exact handle on the expressive power of the basic modal language, but also that its bisimulation analysis can be generalized, extended, and exported in many directions [51]. Moreover, it can be used to map out the *relative* expressive power of languages by finding properties that are expressible in one language, but not expressible in another (which is where bisimulations come in).

Bisimulation-based analyses can actually be pushed down from the whole language to the specific operators and constructs that a language admits. For instance, as we will shortly see, in certain description logics, the $\Diamond$-operator is only allowed to have the constant $\top$ ('true' or 'verum') as its argument; by dropping the clause 'that is Z-related to v_1' at the end of item 2 of Definition 1, we get a new notion of bisimulation that precisely characterizes this restricted modal language. Similarly, the semantic contribution of the disjunction $\vee$ can be captured by changing bisimulations from relations that link objects to objects into relations that link objects to sets of objects [42].

This observation becomes especially useful when thinking about the expressive power of so-called description logics. Description logics [18] are a collection of formal languages originating from the area of Knowledge Representation in artificial intelligence. They split the available information about a situation into two parts, global and local, using a 'T-Box' and an 'A-Box.' The T-Box contains *terminological* information: definitions of the basic and derived notions and of the ways they are related. The A-Box contains *assertional* information: specific information about particular individuals. The formulas put in these boxes are built up from 'concepts' (denoting sets of objects) and 'roles' (denoting binary relations between objects). Description logics differ in the constructions they admit for building complex concepts and roles. For instance, the well-known description logic $\mathcal{FL}^-$ has the following ingredients:

- atomic concepts (which single out sets of objects that share a property),
- two constant concepts $\top$ ('top', the set of all objects) and $\bot$ ('bottom', the empty set),
- universal quantification of the form $\forall R.C$ (which stands for the set of objects d such that any e that is related to d via the relation R, has the property C),
- conjunctions of concepts $C \sqcap D$ (which stands for the set of objects satisfying both C and D), and
- unqualified existential quantification of the form $\exists R.\top$ (which stands for the set of objects d that are R-related to some object).

The description logic $\mathcal{AL}$ extends $\mathcal{FL}^-$ by allowing negations of atomic concepts; and $\mathcal{ALC}$ extends $\mathcal{AL}$ by allowing arbitrary negations.

Put somewhat loosely, description logics are modal logics with added facilities for the structured representation of information. In particular, the concepts that can be constructed in $\mathcal{ALC}$ correspond directly and precisely to formulas of multi-modal logic.

This close connection between modal and description logic has enabled the transfer of tools and techniques between the two fields. To give but one example, the earlier bisimulation-based analysis of ordinary modal logic can be extended to characterize and compare the expressive power of description logics: map description logics into first-order logic, and characterize the resulting fragments in terms of suitable notions of bisimulation. Figure 2, taken from [42], contains a complete classification of the relative expressive power of the so-called $\mathcal{AL}$- and $\mathcal{FL}^-$-hierarchies of description logics. These results are currently being ported to the restricted setting of finite models [36], and extensions that allow one to cope with the earlier A-Boxes and T-Boxes, have been discussed by Areces and de Rijke [3, 4].

A Mismatch? So far, we've seen a classification of the expressive power of a family of description logics. But what do we know about the computational costs of reasoning in these logics? Given our focus on the balance between inference and representation, we'd like to know to which extent differences in expressive power are mirrored by differences in computational costs.

The strong emphasis on reasoning methods in the description logic community has given rise to a wide range of complexity-theoretic results. In

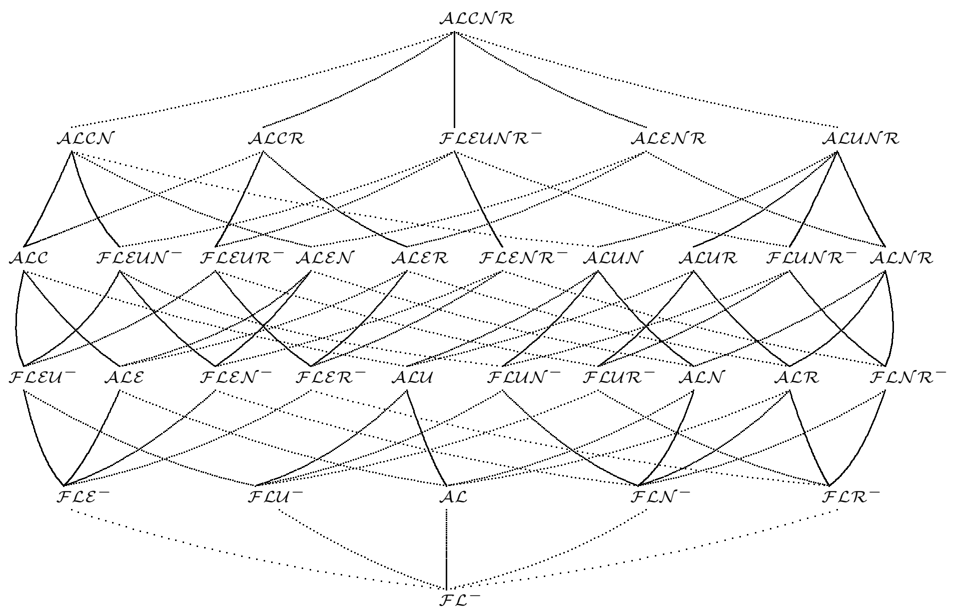


Figure 2: Classifying expressive power.

particular, for all of the languages mentioned in Figure 2, the computational complexity of the satisfiability problem is known [19]. These complexity results can be coupled to the expressivity classification given in Figure 2, or can they? Let's have a look:

- The satisfiability problems for $\mathcal{AL}$ and $\mathcal{ALN}$ are both decidable in polynomial time, but according to our analysis $\mathcal{ALN}$ is strictly more expressive than $\mathcal{AL}$.

- The satisfiability problems for $\mathcal{ALE}$, $\mathcal{ALR}$, and $\mathcal{ALER}$ are all NP-complete, yet $\mathcal{ALER}$ is the most expressive of these.

- The satisfiability problems for $\mathcal{ALU}$ and $\mathcal{ALUN}$ are coNP-complete, but $\mathcal{ALUN}$ is strictly more expressive than $\mathcal{ALU}$.

- The satisfiability problems for $\mathcal{ALC}$, $\mathcal{ALUR}$, $\mathcal{ALNR}$, $\mathcal{ALCN}$, $\mathcal{ALCR}$, $\mathcal{ALEN}$, $\mathcal{ALENR}$, $\mathcal{ALUNR}$, and $\mathcal{ALCNR}$ are all PSPACE-complete, yet the logic $\mathcal{ALCNR}$ is the most expressive of these.

What is the upshot? Description logics whose satisfiability problems are complete for the same complexity class need not have the same expressive power in our sense. There are two sides to this. First, at equal computational costs one may wish to opt for the most expressive logics — can we perhaps identify cut-off points where any further extensions to the expressive power pushes the complexity to a higher class?. Second, what is the precise relation between these alternative ways of classifying description logics? Should we compare model-theoretic classifications of the expressive power of description logics to the complexity of other reasoning tasks? To model checking perhaps? Or should we look at average-case complexity instead of worst-case complexity?

Experimental Aspects: The Random Modal QBF Test Set

As we've just seen, theoretical investigations into the balance between inference and representation naturally give rise to experimental issues: determining any useful average-case complexity by theoretical means is essentially impossible... In the area of propositional satisfiability checking there is large body of experimental knowledge; see e.g., [25] for a recent showcase. In contrast, empirical aspects of modal satisfiability checking have only recently drawn the attention of researchers. We now have a number of test sets, some of which have been evaluated extensively [6, 31, 26, 35, 34]. In addition, we have a clear set of guidelines for performing empirical testing in the setting of modal logic [31, 34].

Below we report on an empirical evaluation of one of the test sets that is currently in use for evaluating modal satisfiability solvers, viz. the random modal QBF test set. The first random generation technique used in testing modal decision procedures, the random $3\mathrm{CNF}_{\Box_m}$ test methodology, was proposed in [26]; its subsequent development is described in [34]. The random modal quantified Boolean formula (QBF) test set was proposed by Massacci [45], and used in the 1999 and 2000 editions of the TANCS system comparisons [65]. It is based on the idea of randomly generating QBFs and then translating these into modal logic. Let us explain these two steps in more detail.

Generating QBFs. QBFs have the following shape: $\mathsf{Q}_1 v_1 \dots \mathsf{Q}_n v_n\, \mathit{CNF}(v_1, \dots, v_n)$, where each Q is either $\forall$ or $\exists$. That is, QBFs are prenex formulas built up from proposition letters, using the booleans, and $\forall v\, \beta$ and $\exists v\, \beta$ (where v is any proposition letter).

What is involved in evaluating a QBF? We start by peeling off the outermost quantifier; if it's $\exists v$, we choose one of the truth values 1 or 0 and substitute for the newly freed occurrence of v; if it's $\forall v$, substitute both 1 and 0 for the newly freed occurrences of v. In short, while evaluating QBFs we are generating a tree, where existential quantifiers increase the depth, and universal quantifiers force branching.

In the *random modal QBF test set*, 4 parameters play a role: c, d, v, k:

- c is the number of clauses of the randomly generated QBF;
- d is the alternation depth of the randomly generated QBF; it is *not* the modal depth of the modal translation (more on this below);
- v is the number of variables used per alternation;
- and k is the number of different variables used per clause.

The *QBF-validity problem* is the problem of deciding whether a QBF without free variables is valid; it is known to be PSPACE-complete.

Here's a concrete example. Using $d = 3$ and $v = 4$ we can generate

$$\underbrace{\forall \overbrace{v_{34}v_{33}v_{32}v_{31}}^{4} \exists \overbrace{v_{24}v_{23}v_{22}v_{21}}^{4} \forall \overbrace{v_{14}v_{13}v_{12}v_{11}}^{4} \exists \overbrace{v_{04}v_{03}v_{02}v_{01}}^{4}}_{3} \ \mathit{CNF}(v_{01}, \ldots, v_{34}).$$

Each clause in $\mathit{CNF}(v_{01}, \ldots, v_{34})$ has k different variables (default 4) and each is negated with a certain probability (default 0.5). The first and the third variable (if it exists) are existentially quantified. The second and fourth variable are universally quantified. This aims at eliminating trivially unsatisfiable formulas. Other literals are either universally or existentially quantified variables with a certain probability (default 0.5). The depth of each literal is randomly chosen from 1 to d.

By increasing the parameter d from odd to even, a layer of existential quantifiers is added at the beginning of the formula, and, conversely, when d increases from even to odd, a layer of universal quantifiers is added.

Translating into Modal Logic. The QBF that is produced by the random generator is translated into the basic modal logic with the usual boolean operators and $\Box$, $\Diamond$, using a variant of an encoding due to Ladner [43]. The core idea is to capture, by means of a modal formula, the 'peel off quantifiers and substitute' evaluation process for a given input QBF. The translation forces branching in the structure of the possible model whenever a universal quantifier is found in the original formula; it keeps the branches separate, and makes sure there are enough modal levels in the model. It forces the structure of the possible model to be a tree, and the resulting formula is satisfiable iff the original formula is.

Some Test Results. With these preliminaries out of the way, we can report on some test results due to Heguiabehere and de Rijke [30]. To evaluate the QBF test set, we used 3 satisfiability solvers for modal logic. First, we used the general first-order prover SPASS [61], version 1.0.3, extended with the layered translation of modal formulas into first-order formulas as presented in [5]. Second, we used MSPASS version V 1.0.0t.1.2.a [50]. And, third, we used *SAT version 1.3 [64]. To facilitate future comparisons we used as many default settings as possible; for details on the settings, please consult [30].

We generated 64 instances of each problem, and the outputs of the generator were translated to the formats of the provers being used; in one case we had to convert modal formulas to first-order logic formulas. The resulting file sizes were linear in c, even though the linear coefficient varied from one solver to another.

Our main measurements concerned both CPU time elapsed (with a 10800 second timeout) and a time independent measure: the number of clauses generated for SPASS and for MSPASS, and the number of unit propagations for *SAT.

We ran a large number of sweeps with each of the three provers, with $v = 2$ and increasing d from 1 to 4 (and to 5 in the case of *SAT), while increasing c from 1 to 100. The resulting CPU times and the number of clauses generated/unit propagations are depicted in Figure 3; the curves for $d = 1$, $d = 2$ do not extend to the right-hand side of the plots, as the formulas being generated with these settings are simply too small to be able to accommodate larger numbers of clauses. Many things are worth noting about Figure 3; due to lack of space, we only mention one: the shape of the curves is strongly dependent on the solver used.

Horrocks, Patel-Schneider, and Sebastiani [34] have put forward a long list of general criteria for evaluating modal test methodologies. In essence, they boil down to demanding a reproducible sample of an interesting portion of the input space with appropriate difficulty. We briefly discuss some of these criteria as they relate to the modal QBF test set. Clearly, *reproducibility* is guaranteed for the modal QBF test set. However, the modal QBF test set only seems to represent a restricted area of the whole input space; that is, it scores low on *representativeness.* There are three reasons for this. First, the QBF test set provides poor coverage of the satisfiable region, and especially of the easily satisfiable region; most of the modally encoded QBF-formulas generated with values of v and d that are within reach of today's tools, are hard and unsatisfiable, as suggested by Figure 4.

Second, the modally encoded QBFs are of a very special shape, which

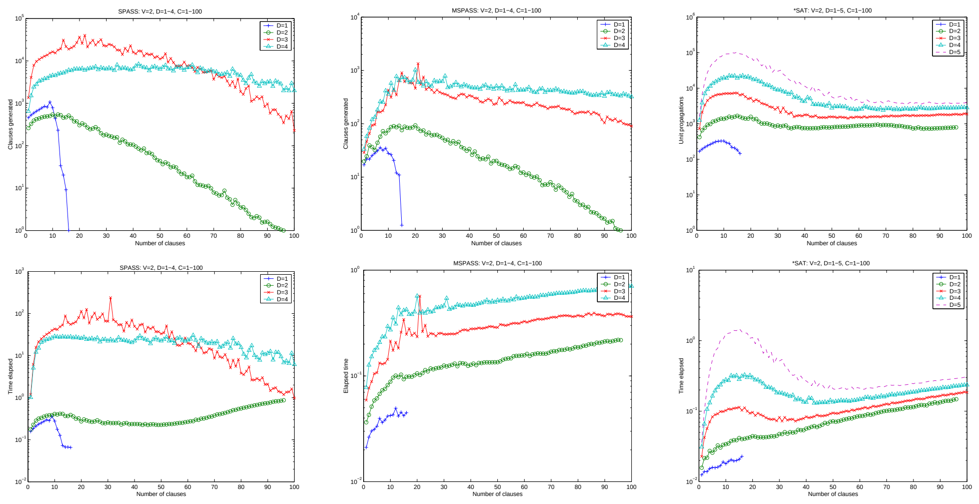


Figure 3: Results on QBF test sets, $v = 2$, $d = 1$–4 (5), 64 samples/point, mean values. (Top): clauses generated/unit propagations, log scale. (Bottom): CPU time in seconds, log scale. (Left): SPASS. (Middle): MSPASS. (Right): *SAT.

seems to lead to the so-called staircase phenomenon for some solvers; see Figure 3 (Left), where curves cross as the d parameter is increased, both for CPU times and clauses generated. And third, the v and d parameters end up being substantially overlapping and interrelated as part of the translation of QBFs into modal formulas. A strong point in favor of the QBF test set is that it is possible to generate hard problems with a large modal depth which are still within reach of today's modal satisfiability solvers; in this respect the QBF random test methodology fares better than the New_3CNF$_{\Box_m}$ test methodology, as reported in [34].

The QBF test set scores low on the *satisfiable vs. unsatisfiable balance*: it has a strong preference for generating unsatisfiable instances. The levels of *difficulty* offered by the test set are sufficient, as they range from next to trivial to too hard for today's systems. The tests *terminate* and provide information in a reasonable amount of time. But, the set suffers from *oversize*, which forces one to consider both time dependent and time independent measurements for determining the performance of solvers.

Summarizing, the random modal QBF test methodology provides useful test sets that should, however, not be used as the sole measure in the evaluation of modal satisfiability solvers. In particular, the standardized tests provided by TANCS (with $c = 20$ and $v = d = 2$) do not provide informative measurements.

There's a lot more to say about the modal QBF test set. For a start, we are currently running the test set with provers other than the three considered here. Moreover, plans for further work include explorations beyond the default settings (more literals, different encodings of QBFs into modal problems, alternative settings for the various probabilities), as well as systematic comparisons with other test methodologies. An interesting (but more abstract) question is if and how the measurements produced by a structured problem such as the modal QBF test set are qualitatively different from results obtained with unstructured randomly generated problems.

3 Reasoning and Representing

While evaluations of satisfiability solvers — propositional, modal, or otherwise — are interesting and essential for determining the quality of reasoning components, they are not completely satisfactory for our purposes: understanding the interplay between inference and representation. Obviously, to make progress on our core theme we need to consider both inference and

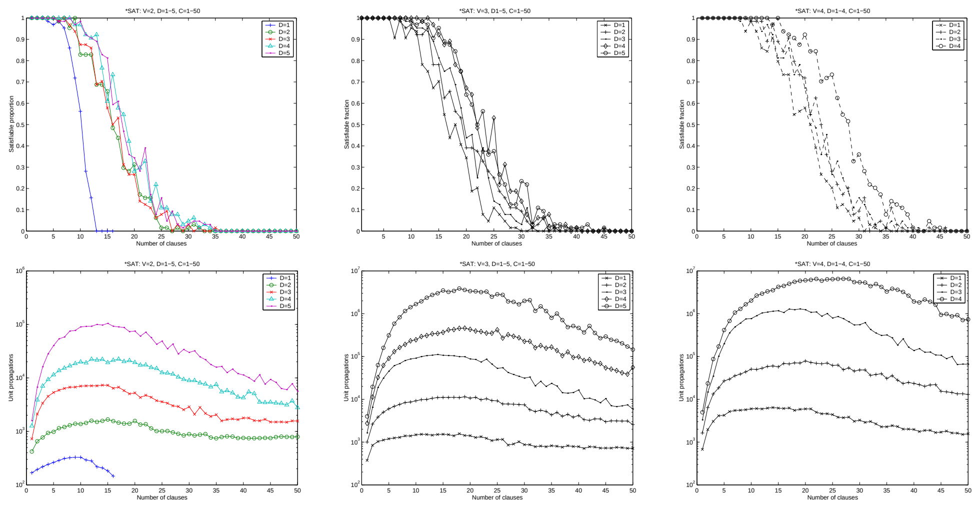


Figure 4: *SAT results for d = 1–4 (5), c = 1–50, 64 samples/point, mean values. (Top): Satisfiable fraction. (Bottom): Unit propagations. (Left): $v = 2$. (Middle): $v = 3$. (Right): $v = 4$.

the representations on which inference acts.

Before looking at look at experimental tasks that involve both inference and representation, let's briefly explore the scientific background: *computational semantics* [57]. An new textbook takes 'computational semantics' to mean the business of using a computer to actually build logical representations of fragments of natural language (*semantic construction*) and reason with the result (*inference*) [10].

It's fair to say that computational semantics is characterized by a strong emphasis on theoretical analyses, both linguistically, logically, and computationally; Van Eijck's chapter provides a very nice example of this. In other areas of natural language understanding, developments have gone in the opposite direction. Indeed, experimental evaluation has excited a great deal of interest in the language engineering world as of late [23, 32, 60]. Not only do we want to know which programs perform best, but also, the developers of a program want to know when modifications improve performance, and how much, and what combinations of modifications are optimal.

We will now take a look at two test sets for computational semantics, one dates back to the mid 1990s, the other is currently being developed.

The FraCaS Test Suite

The FraCaS (A Framework for Computational Semantics) project was a European project that ran during the mid 1990s [21]. It was aimed at bringing together the various approaches to computational semantics that existed at the time, and at investigating their potential for applications. (Incidentally, one of the Spinoza project members, Jan van Eijck) was leader of the Dutch site that took part in FraCaS.)

As part of the FraCaS project, a semantic test suite has been developed [16]. The test suite is meant to be a basis for a useful and theory/system independent semantic tool. Inference tasks are seen as the best way of testing a the semantic capacity of a natural language processing system, but while the use of random and/or (modified) past inputs has become the standard practice for testing and evaluating automated reasoning techniques and optimizations, for controlled experiments in inference and representation we need test sets with the following three key ingredients:

1. a document collection from which to extract representations, as well an explicit and detailed definition of the task to be carried out with these representations,

2. a 'gold standard' corpus of correct answers, so it is possible to say how much of the time a program gets it right, and

3. a baseline, i.e., a 'minimal' or 'maximal' score against which the performance of competing methods can be compared.

The document collection and task definition in the FraCaS test suite take the following form: the system is given some natural language input T, usually consisting of a very small number of sentences, and is then presented with a claim S and asked to determine whether S follows from T. Since tests should be neutral as regards the semantic representation formalisms they use, the test claim S is offered in the form of a natural language sentence (like the input T).

Inference tasks of the form just described form a continuum, ranging from formalizing an argument and deriving the conclusion from the premises to the kind of text analysis that is often taught in high schools, where a student is presented with a text and then with certain questions about it. The test suite contains both valid and invalid inferences, loosely grouped together into various linguistic and semantic phenomena. These include generalized quantifiers, plurals, (nominal) anaphora, ellipsis, adjectives, comparatives, temporal reference, verbs, and attitudes. Here are a few examples.

(3.81) Smith, Jones, and Anderson signed the contract.
____
Did Jones sign the contract?

[Yes]

(3.334) Smith hoped that ITEL had won the contract in 1992.
____
Did ITEL win the contract in 1992?

[Don't know]

The numbers refer to the numbers of these examples in the FraCaS test suite, which contains over 3000 inference tasks like these. The comment in square brackets indicates the correct answer; nearly all of the answers to the tasks in the test suite are amongst the following: 'yes', 'no', 'don't know', 'yes, on one reading'.

Unfortunately, the FraCaS test suite does not come with a baseline score, thus making it hard to use the suite for comparison between approaches. Moreover, as far as we are aware, only a very small part of the test suite ($< 5\%$) has actually been used for evaluation purposes [63].

Towards a Test Set for Computational Semantics

We now report on an ongoing effort aimed at devising a test set that allows us to evaluate inference in computational semantics. The aim of the test set is to allow us to identify good notions of inference for computational semantics.

Lots of recent work on inference in computational semantics has focused on so-called *intra-document* entailment checking, where one tries to find and exploit inferential links within a single document. *Informativity checking* has been one of the most popular instances; informativity is an often used notion in natural language processing and understanding, and Blackburn et al. [11] show that informativity can be treated as an entailment problem: a piece of information NEW is informative with respect to a discourse context OLD and general world knowledge KB just in case the implication $\text{OLD} \wedge \text{KB} \rightarrow \text{NEW}$ is not valid. Gardent and Webber [24] show how informativity may be used in the discourse interpretation of phenomena such as noun-noun compounds, metonymy, and definite noun phrases.

In contrast, the test set that we are about to describe focuses on *inter-document* entailment checking between segments of natural language text taken from (possibly) different documents; the example and real-world tasks that were mentioned in Section 1 provide settings in which this flavor of inference can be found, either implicitly or explicitly.

As we pointed out above, for controlled experiments in inference and representation we need test sets with the following key ingredients: a document collection, a gold standard, and a baseline score. Let's take a brief look at these ingredients for a test set that is being set up by Monz and de Rijke [46] within the *Logic in Action* project. The document collection is a small corpus consisting of (open domain) news stories from AP news wire, BBC, CNN, L.A. Times, Reuters, USA Today, Washington Post, Washington Times. At present the collection consists of 98 documents organized in 30 topics; this was done by hand. All documents belonging to a single topic were released on the same day and describe the same event; see Figure 1 for two documents from Topic 6. The documents were segmented into paragraphs; in news stories these tend to be short, and we found that they rarely exceed 4 sentences. For preliminary evaluation purposes, only Topics 1–21 were used; see Table 1 for some figures.

Let's consider the gold standard now. Its aim is to create *ideal* entailment judgments for comparison with automatically generated ones. For each topic, 2 documents were selected at random; at present (summer 2001) a human subject has determined all entailment relations between segments

Table 1: Statistics on Topics 1–21, 69 documents.

	average per topic	
number of documents	3.3	docs.
document length	612	words
total length of documents	2115	words
length of longest document	783	words
length of shortest document	444	words
segments per document	16.4	
total number of segments	55.9	

in different documents (but within the same topic). Judgments were made on a scale 0–2, according to the extent to which one segment was found to entail another. More specifically, for pairs of segments $s_{i,d}$ in document d and $s_{j,d'}$ in document d', the pair $(s_{i,d}, s_{j,d'})$ was to be assigned a score of 2 if everything expressed by the second segment $(s_{j,d'})$ is already expressed by the first segment $(s_{i,d})$. A score of 1 was to be assigned if the contents of an important subsegment of $s_{j,d'}$ is expressed by the first segment, $s_{i,d}$. In all other cases the pair $(s_{i,d}, s_{j,d'})$ was to be assigned a 0. Out of 12083 *potential entailment pairs*, 501 (4.15%) received a score of 1, and only 89 (0.73%) received a score of 2.

Providing a baseline score requires a couple of things. First of all, a method for computing entailments between text segments. We used a computationally efficient method that works on very minimal representations that can easily be generated for arbitrary domains, and that provides graded outcomes instead of strict yes/no answers. It is based on a familiar similarity measure from information retrieval. For a given topic, let N denote the total number text segments in the topic, and n_i denotes the number of segments in which a given term t_i occurs. The *weight* of term t_i within a given topic is its *inverse document frequency*:

$$idf_i = \log\left(\frac{N}{n_i}\right).$$

Terms that occur in many segments have a lower score than terms that occur in only a few. The intuition is that terms with a higher *idf*-score are better suited for discriminating between segments in a topic; the log is only used to dampen differences between the weights.

We now explain how to compute the *entailment score* $entscore(s_{i,d}, s_{j,d'})$ of two segments $s_{i,d}$ (in document d) and $s_{j,d'}$ (in document d'). The idea is this: how much of the content of $s_{j,d'}$ is already in $s_{i,d}$? Put differently: how many of the content-bearing terms in $s_{j,d'}$ occur in $s_{i,d}$? To answer this question, we compare the sum of the weights of terms that appear in both segments to the sum of the weights of all terms in $s_{j,d'}$:

$$entscore(s_{i,d}, s_{j,d'}) = \frac{\Sigma_{t_k \in (s_{i,d} \cap s_{j,d'})} idf_k}{\Sigma_{t_k \in s_{j,d'}} idf_k}.$$

Clearly, *entscore* varies from 0 to 1. Moreover, *entscore* is not a notion of similarity: in general, $entscore(s_{i,d}, s_{j,d'}) \neq entscore(s_{j,d'}, s_{i,d})$. Now, to work with *entscore* and conclude that $s_{i,d}$ entails $s_{j,d'}$ we need a positive *entailment threshold*: some value $\lambda \in (0, 1]$ such that if $entscore(s_{i,d}, s_{j,d'}) \geq \lambda$ then $s_{i,d}$ entails $s_{j,d'}$, otherwise it does not.

What else do we need to establish a baseline score? Well, we need to assess *entscore*, so that other methods can be compared against it. Some terminology: a *correct* entailment pair is a pair $(s_{i,d}, s_{j,d'})$ for which $s_{i,d}$ does indeed entail $s_{j,d'}$ according to the gold standard. And a *computed* entailment pair is a pair $(s_{i,d}, s_{j,d'})$ for which *entscore* has produced a value above the entailment threshold. Now, for the important evaluation measures. We use *precision* to determine the accuracy of the entailment checking method:

$$\text{Precision} = \frac{\text{\# of correct entailment pairs computed}}{\text{total \# of entailment pairs computed}}.$$

We use *recall* to measure the extent to which our entailment checking method is exhaustive:

$$\text{Recall} = \frac{\text{\# of correct entailment pairs computed}}{\text{total \# of correct entailment pairs}}.$$

Note that both precision and recall depend on the entailment threshold being used: with a low entailment threshold, expect a larger number of computed entailment pairs, and hence a larger number of correctly computed entailment pairs; that is, with a low entailment score, recall will increase.

Using human judgments for two selected documents per topic, we computed average recall and precision values at 11 different entailment thresholds, ranging from 0 to 1, with .1 increments, and averaged over all topics. We used a human rating of either > 0 or of > 1 to classify an entailment pair as correct. The resulting plots are given in Figure 5.
As expected, precision is higher when human judgments > 0 are used to determine the correct entailment pairs than with human judgments > 1. Initially, precision increases as the threshold is increased, but there are drops:

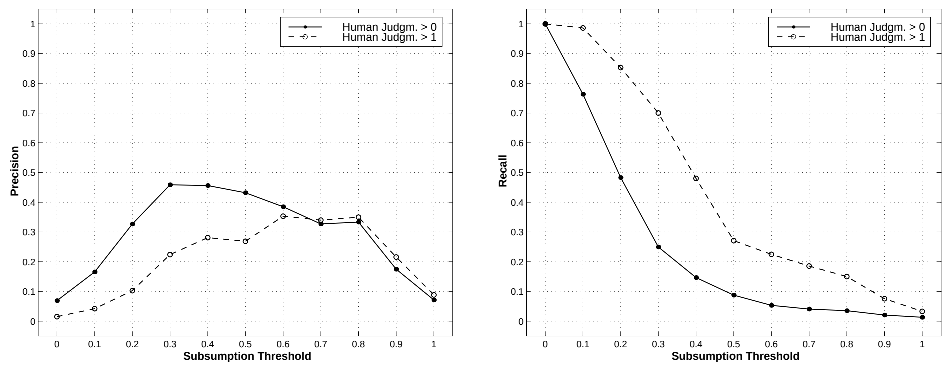


Figure 5: (Left): Average precision with human judgments > 0 and > 1. (Right): Average recall with human judgments > 0 and > 1.

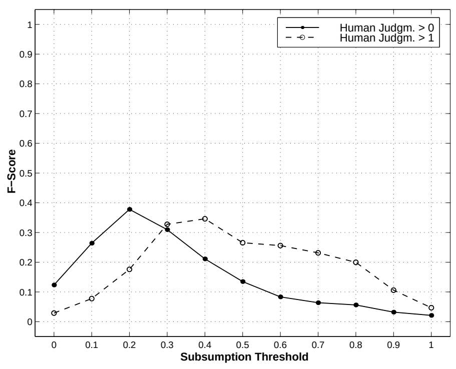
for some topics the threshold is higher than the maximum entailment score, so precision drops to 0. As to recall, as expected, recall is higher when human judgments > 1 are used than with human judgments > 0.

Since precision and recall suggest two different optimal entailment thresholds, there is an obvious question: what is the optimal threshold if precision and recall are equally important? In information retrieval, various methods have been suggested for generating a single performance number, which combines precision and recall aspects, to quantify the usefulness of retrieval methods. One of these is the *harmonic mean* F of recall and precision [56] which is computed as

$$F = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}.$$

The F-score assumes a high value only when both recall and precision are high. We have plotted the average F-scores in Figure 6 (Left). The optimal entailment threshold for human judgments > 0 seems to be around 0.2, and approximately 0.4 for human judgments > 1. This conforms to the intuition that a higher entailment threshold is more effective when human judgments are stricter.

In Figure 6 (Right) we have plotted F-scores per topic, with human judgments > 0. The average over these curves corresponds to the solid line in (Left), and, indeed, the shape of the solid line in (Left) can be recognized in (Right). Note that there is some variance between the topics, and that there are some clear outliers, such as Topic 1 and Topic 2.



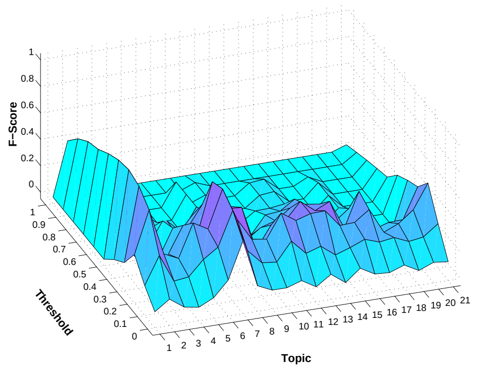


Figure 6: (Left): Average F-scores with human judgments > 0 and > 1. (Right): F-scores across topics, with human judgments > 0.

The F-score suggests that 0.2 (0.4) is to be taken as entailment threshold for identifying entailing segments with a human judgment score of at least 1 (2). This results in an overall precision of 0.33 (0.28) and an overall recall of 0.48 (0.48). Precision and recall figures in the 30% and 40% range are not optimal, but computing entailment is a hard task, and relatively poor results should not come as a surprise. But, even when working with extremely simple representations, our experiments indicate that almost half of all entailment relations can be identified! Identifying only a third of them correctly may seem unsatisfactory, but, again, entailment relations are very sparse.

Summarizing, we have described the development of a test set for computational semantics; the main task for whose evaluation this test set is designed, is computing entailment relations between (open domain) text segments. With this test set (and the baseline scores that it comes with) it is now possible to evaluate computational semantic systems on realistic open domain texts. In particular — and returning to our main theme —, one can start investigating the extent to which deeper levels of semantic analysis yield better recall or precision scores. In ongoing work, Hammarstrom [27] is exploring the use of bag-of-words representations similar to the ones used for obtaining the baseline; for instance, stopword lists and the use of various knowledge sources such as WordNet [68] are being explored. In addition, Hammarstrom is investigating argument structures that have been obtained by partially parsing the input documents, to compute entailment relations.

4 Real-World Tasks

The evaluation work on which we reported in the previous section is subject to a number of criticisms. One objection is that quantitative evaluation discourages novel approaches and risk taking. But this is unlikely to apply as long as reputations do not depend on the outcome. A second objection is that setting up the evaluations, and participating in them is a lot of work; this may well be the case, even for a small-scale scenario such as the one described here, and we are hopeful that it does not create a competitive (as opposed to collaborative) attitude, but, instead, that working on the same problem will prove productive [38].

There is, however, a more serious objection to the empirical evaluation efforts such as the ones described in Section 3: unless the tasks are carefully chosen, they will draw energy away from the fundamental problems in the field. Is computing entailment relations between text segments a useful semantic task? It can clearly be identified as a separate task, but if we are trying to understand the balance between inference and representation, how much do the intermediate outputs that we get from computing entailment relations tell us? How much do they tell us about the impact of deeper representations on the overall performance for various natural language processing applications?

Information retrieval (IR) is a set of techniques that serves the primary purpose of finding documents that are relevant to an information need [8]. In other words, IR is concerned with accessing the *content* of documents, which might lead one to believe that this is a real-world task with a clear potential for natural language processing (NLP), and, in particular, for semantics. For years, researchers in NLP have attempted to use increasingly sophisticated levels of linguistic analysis to improve the quality of IR methods [62]. However, most NLP methods do not improve IR effectiveness, and years of experimental evaluation (of mainly English queries and English documents) have lead some to claim that, in IR, NLP errors hurt more than NLP techniques help [67].

Is there no place whatsoever for NLP in IR? Of course not! Very basic NLP is very common in IR: think of tokenizing, stopping, stemming. And more advanced but fairly robust NLP is also common, in the form of phrase identification and named entity extraction. But truly advanced NLP is problematic. It is likely to be more useful for tasks other than strict document retrieval. In the remainder of this section we take a brief look at two things: the importance of low-level NLP tools for IR, and a recent development in IR research that has lead to the formulation of a task where advanced NLP does make a difference.

Non-English Information Retrieval

The Cross-Language Evaluation Forum (CLEF, [14]) aims at promoting research in cross-language IR by providing an infrastructure for the testing and evaluation of IR systems operating on European languages, and by creating test suites of reusable data which can be employed by researchers for benchmarking purposes. These objectives are being addressed through the organization of a series of annual system evaluation events. CLEF 2001 was the second event in this series, and it offered four main evaluation tracks: multi-lingual, bilingual, and monolingual (non-English) retrieval, as well as domain-specific system evaluation.

The **Derive** project [17] is one of the spin-offs of the *Logic in Action* project. Monz and de Rijke, the principal investigators in **Derive**, took part in the CLEF 2001 evaluation in three monolingual tasks: Dutch, German, and Italian [47]. We were particularly interested in the effects of shallow morphological analyses: stemming or lemmatization, and compound splitting. For the purposes of this chapter, we restrict ourselves to a brief description of our results for Dutch and German.

All submitted runs used FlexIR, an information retrieval system developed by Christof Monz [47]. The main goal underlying FlexIR's design is to facilitate flexible experimentation with a variety of retrieval components and techniques. FlexIR is implemented in Perl; it is built around the standard UNIX pipeline architecture, and supports many types of scoring, indexing, and retrieval tools.

The retrieval model underlying FlexIR is the standard vector space model [8]. All our official runs for CLEF 2001 used the Lnu.ltc weighting scheme [13] to compute the similarity between a query and a document. Blind feedback was applied to expand the original query with related terms; term weights were recomputed by using the standard Rocchio method [54], where we considered the top 10 documents to be relevant and the bottom 250 documents to be non-relevant. We allowed at most 20 terms to be added to the original query.

Inflectional Morphology. Previous retrieval experimentation [22] in English did not show consistent improvements by applying morphological normalization such as rule-based stemming [52] or lexical stemming [29]. For languages that are morphologically richer than English, such as Dutch, German, Italian or Spanish, we have a similar mixed picture from CLEF 2000 and other experiments. Kraaij and Pohlmann [40] report that for Dutch the effect of stemming on on retrieval performance is limited; it tends to help as many queries as it hurts. For German and French, the results are similar to those for English [49].

Although versions of Porter's stemmer are available for both Dutch and German, we decided to use a lexical-based stemmer, or lemmatizer, because it tends to be less aggressive than rule-based stemmers, and we conjectured that this might benefit further morphological analyses such as compound splitting (see below). The lemmatizer is part of the TreeTagger part-of-speech tagger [55]. Each word is assigned its syntactic root by lexical look-up. Number, case, and tense information is removed, leaving other morphological processes such as nominalization intact. As an example in German, *Vereinbarung* (English: agreement) and German: *vereinbaren* (English: agree) are not conflated.

Compound Splitting. Compound splitting is not an issue in English since almost all compounds, such as *Computer Science*, *peace agreement*, etc. are separated by a white space, disregarding some exceptions such as *database* or *bookshelf*. In Dutch and German compounds are not separated and compound building is a very common phenomenon. Kraaij and Pohlman [41] show that compound splitting leads to a significant improvement of retrieval performance for Dutch, and Moulinier et al. [49] obtain similar results for German.

In some of our runs for Dutch and German we used a compound splitter. Our compound splitter for Dutch was built using the Dutch lexicon provided by Celex [7], while our German compound splitter used the part-of-speech information provided by TreeTagger. We limited ourselves to noun-noun compounds. For instance, the German compound *Friedensvertrag* (English: *peace agreement*) is split into *Frieden*+s *Vertrag*.

For retrieval purposes, each document in the collection is analyzed and if a compound is identified, all of its parts are added to the document. In some cases, compound splitting can give rather awkward results, e.g., German: *trompet* (English: trumpet) is split into *trom* (drum) and *pet* (cap). Whereas 'drum' is semantically related to 'trumpet,' this is not obvious for 'cap.' The current version of our compound splitters are not tuned for retrieval purposes; for instance, we did not try to avoid the addition of unrelated compound parts.

Some Runs. Following the TREC philosophy, information needs at CLEF are called *topics*; each topic consists of three parts: a brief title field; a one-sentence description field; and a more complex narrative field specifying the relevance assessment criteria. At CLEF 2001 a total of 50 topics was used for evaluation purposes.

For both Dutch and German we submitted three types of runs:

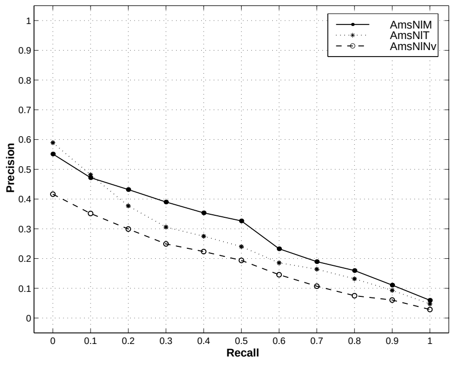
- *M (Morphological)* The title and the description field of the topic are used to generate the retrieval query (this was a mandatory requirement to be met by at least one of the runs). Words are morphologically normalized and compounds are split (Dutch and German). Blind feedback is applied to the top 10 documents adding at most 20 terms to the original query.

- *Nv (Naïve)* The title and the description field of the topic are used to generate the retrieval query. Blind feedback is applied to the top 10 documents adding at most 20 terms to the original query. In contrast to runs of type M, no morphological normalization or compound splitting are applied.

- *T (Title only)* The same retrieval and document processing techniques are used as for runs of type M, but query formulation is restricted to the title field of the topic.

There are several motivations for this set of runs. Type M runs were intended to be the most effective runs, using techniques which are generally believed to improve retrieval effectiveness. Type T runs use the same techniques as type M runs, but queries are much shorter and, therefore, more closely resemble queries posed by a non-expert. Type Nv runs were intended as a contrast to type M runs, where no language specific techniques are applied.

Some Results. Let's look at some of the results of our CLEF 2001 submissions. The measures used are *precision* (what fraction of the retrieved documents is relevant) and *recall* (what fraction of the relevant documents has been retrieved), combined into a single so-called precision-recall curve. Figure 7 displays the interpolated precision-recall curves for both Dutch and German, with precision interpolated at the 11 standard recall levels (0%, 10%, ..., 100%). A look at the non-interpolated avg. precisions for type M and type Nv runs (Table 2) reveals that morphological normalization does



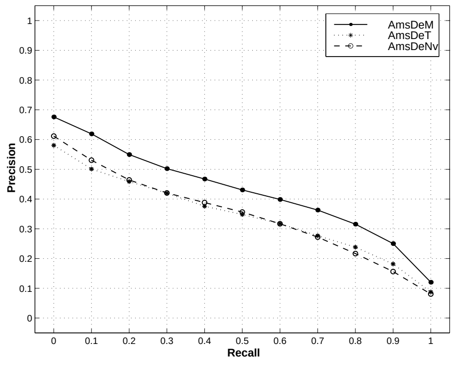


Figure 7: 11pt interpolated avg. precision for all submitted runs. (Left): Dutch. (Right): German.

result in significant improvements[1] in effectiveness: ≈ 25% for German and even ≈ 54% for Dutch. It is not obvious why the improvement for Dutch is so much bigger than for German. One reason could be that our precision scores for Dutch are, in general, considerably lower than for German. Our results suggest that the improvement brought about by compound splitting (plus stemming) is independent from the underlying retrieval engine.

Table 2: Non-interpolated avg. precisions of Type M runs vs. Type Nv runs.

	Dutch	German
Naïve (Nv)	0.1833	0.3342
+ Morphological Analysis (M)	0.2833 (+54.6%)	0.4172 (+24.8%)

Another interesting question is to compare queries that were formulated by using the title and the description field of the topic to queries that were formulated by using the title of the topic only. Queries based on title information only are much shorter and more closely resemble queries a non-expert would ask. Table 3 shows that for both Dutch and German the decrease in effectiveness is certainly significant but not too dramatic.

Table 3: Non-interpolated avg. precisions of TD-queries vs. T-queries.

	Dutch	German
Morphological Analysis (M)	0.2833	0.4172
Title only (T)	0.2418 (−14.6%)	0.3342 (−19.9%)

[1]Note that 'significant improvement' here refers to the definition in [59], where changes of more than 5% are considered significant.

These and other experiments that we have carried out for CLEF 2001 strongly confirm the believe that morphological normalization does improve retrieval effectiveness significantly. Since the morphological analyses carried out here were still rather restricted, it would be interesting to see what impact additional analyses, e.g., stripping off prefixes and recognizing nominalizations, would have.

Question Answering

Low-level natural language processing does have an important role to play in information retrieval, and probably even more so in retrieval with non-English languages. However, to satisfy our interest in understanding the balance between representation and inference, we consider a task where deeper levels of analysis seem appropriate and effective. This task is *question answering*, as introduced in TREC-8.

TREC [66] is short for 'Text REtrieval Conference,' a series of conferences run by NIST and organized by the IR research community that has been going since 1991. The focus was originally on 'large' text collections (multi-Gb), but nowadays it includes many IR-related tracks, including *question answering*. The question answering (QA) track began in 1999, at TREC-8. The QA track is motivated by the following observations. IR typically retrieves or works with documents: the task is to find documents that are relevant, or to rank and group documents on the same topic. However, people often want a sentence fragment or phrase as the answer to their question:

- Who was the first man to set foot on the moon?
- What is the moon made of?
- How many members are in the U.S. Congress?

The QA track signifies a move away from document retrieval and towards answer retrieval.

The documents used in the task consist mostly of newspaper articles and thus contain information on a wide variety of subjects. In TREC-8 (1999) participants were given 200 fact-based, short-answer questions such as those mentioned above. Each question was guaranteed to have at least one document in the collection that explicitly answered the question.

Participating systems returned a ranked list of five strings per question, such that each answer string was believed to contain an answer to the question. Answer strings were limited to either 50 or 250 bytes, and could either be extracted from the corresponding document or automatically generated from information contained in the document.

Human assessors read each string and made binary decisions as to whether the string actually did contain an answer to the question in the context provided by the document. Given a set of judgments for the answer strings, the score computed for a submission was mean reciprocal rank, defined as follows. An individual question received a score of $1/n$, where n is the rank at which the first correct response was returned, or 0 if none of the five responses contained a correct answer. The score of a submission was then the mean of the individual questions' reciprocal ranks.

While the same basic task was performed in the TREC-8 and TREC-9 QA tracks, there were some differences. The set of questions was larger for TREC-9 (693 instead of 200), and the document collection was larger as well (3 Gb instead of 2 Gb). Another difference between the TREC-8 and TREC-9 QA tracks was the use of question variants. The TREC-9 question set contained 500 questions, plus an additional 193 that were syntactic variants of one of the 500. The purpose of the syntactic variants was to investigate whether QA systems are robust to the variety of different ways a question can be phrased.

The TREC-10 QA track consists of three separate tasks, called the main task, the list task, and the context task. The main task is similar to the task in previous QA tracks, and consists of 500 questions, some of which may not have answers in the document collection. The list task consists of 25 questions in the same format as the main task, each of which requires information from more than one document to produce the answer. For example, a list question such as *Name the countries the Pope visited in 1994* requires finding multiple documents that describe the Pope's visits and extracting the country from each. Systems aimed at the list task need to identify duplicate reports of the same visit so that countries are listed only once per visit. The context task is a first attempt to force systems to exploit context when answering questions. An example series of questions is: *Whose band used "Moonlight Serenade" as its theme song?*, *What instrument did he play?*, *Who was the lead trumpeter?*, and *Name another song the band was famous for.*

NLP Applicability. Many participants in the QA tracks in TREC-8 and TREC-9 used a variant of the following general strategy. The system first attempted to classify a question according to the type of its answer as suggested by its question word. For instance, a question that begins with 'where'

(*Where is Belize located?*) implies that a location is being sought. Next the system retrieved a small portion of the document collection using standard document retrieval technology and the question as the query. The system performed a shallow parse of the returned documents to detect entities of the same type as the answer. If an entity of the required type was found sufficiently close to the question's words, the system returned that entity as a response.

This general strategy clearly indicates the potential for the (successful) use of NLP. Let's illustrate this by means of some examples. Consider the following question: *How many calories are there in a Big Mac?* We need to be able to recognize that 'Big Mac' is a phrase or a name, and that the question requires a number as an answer. More generally, as part of the first step in the general strategy we need to detect phrases and recognize named entities (persons, locations, companies, organizations, ...). We also need to classify questions with respect to the type of answer they require, which calls for a combination of (partially) parsing the question and pattern matching.

Moreover, there seems to be room for the use of semantics in question answering. Let's give just two examples, related to anaphora resolution and synonym detection/word-sense disambiguation. Consider the following question from TREC-8: *Who invented the paper clip?* The document collection contained the following answer snippet:

> *The paper clip, weighing a desk-crushing 1,320 pounds, is a faithful copy of Norwegian Johan Vaaler's 1899 invention,... to honor the Norwegian who invented the office helper 90 years ago.*

Clearly, a modicum of anaphora resolution seems essential for question answering; experimental evidence seems to support this [48]. As to synonym detection/word-sense disambiguation, consider the following question: *Who invented the electric guitar?* The only relevant snippet in the document collection is

> *Adolph Rickenbacker was a Swiss immigrant who patented the first solid-body electric guitar.*

A very strict matching between the argument structure derived from the question and the one derived for the text snippet above will not result in a match: one should allow for partial matches and for matches with synonyms or even hyponyms. WordNet [68] is the obvious resource for the relevant information, but as many experiments in IR have shown, without disambiguation the use of WordNet can lead to significant topic drift.

The University of Amsterdam at TREC-10. We took part in the QA track in TREC-10 (2001) using a variant of the general strategy outlined above; see Figure 8 for a high-level overview of the system.

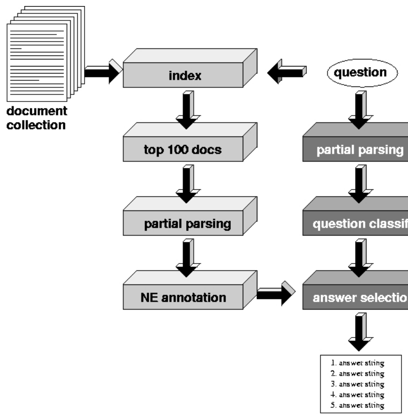


Figure 8: Architecture of the University of Amsterdam's question answering system for TREC-10.

We invested considerable effort in making sure that the initial document retrieval step was of high quality. To this end the FlexIR system mentioned was fine-tuned; results from TREC-9 indicated that this is helpful. We used WordNet [68] as a source of related words for the initial query and as a means of determining whether an entity extracted from a passage matched the required answer type. We used comparatively simple answer extraction and selection techniques.

This was our first participation in TREC. True to the Olympic spirit, our main aim was to take part and set a baseline on which to build and improve in future years. At the time of writing, our scores are not available yet.

2002 and Beyond. An ambitious roadmap for question answering research was recently developed; it describes a program aimed at increasing the complexity of the types of questions that can be answered, the diversity of sources from which the answers can be drawn, and the means by which answers are displayed [28]. The roadmap also includes a five year plan for introducing aspects of these research issues as subtasks of the TREC question answering track. The QA track in TREC-10 includes the first steps of this roadmap, in the form of the list task and context task mentioned above.

The question answering track provides a very attractive setting for experimenting with varying levels of analysis and for exploring the balance between inference and representation. Our question answering plans for the immediate future involve more sophisticated question classification, coreference resolution, and inference-based answer justification.

5 Conclusion

In this chapter we have offered a glimpse at recent computational work within the *Logic in Action* project and its follow-ups. The focus was on computational aspects of logic, language and communication, with a special interest in the balance between inference and representation. While we did not ignore theoretical analyses, our emphasis was on experimental evaluation.

Evaluation is itself a first-class research activity: creating effective evaluation methods drives rapid progress and better communication within a research community [32]. Experience in competitive evaluations for both automated reasoning tasks (such as satisfiability testing in propositional logic or theorem proving in modal or first-order logic) as well as for natural language processing tasks (such as speech recognition, dialogue systems, information retrieval and information extraction) has been that the focus provided by an evaluation brings research communities together, forces consensus on what is critical about the field, and leads to the development of common resources, all of which then stimulates further progress [38].

Many of the more computationally oriented follow-ups to the Spinoza project *Logic in Action* have a strong evaluation component. This is especially true for the Pionier project *Computing with Meaning* [15] which will be lead by the present author and which is in its start-up phase at the time of writing. The focus of the project is on the very theme that featured so prominently in the present chapter: balancing inference and representation in dealing with natural language texts. The theme will be pursued along a number of dimensions: logically (amongst others, by studying at expressive power and computational complexity, as illustrated in Section 2), computationally (amongst others, by developing and analyzing tests sets and benchmarks as illustrated in Section 3), and from a natural language processing point of view (amongst other, by further developing the kind question answering systems described in Section 4).

A final remark: while this chapter and the *Computing with Meaning* project have a strong emphasis on single-agent communication (i.e., the chapter mostly deals with a single agent coping with textual information), there are obvious multi-agent settings where the ideas discussed in this chapter can be put to good use. Promising examples include *dialogue management* and *understanding systems* as exemplified by the SIRIdUS project [58] and *computer games* as exemplified by Koller's recent work [39].

Acknowledgments

The research reported here is based on joint work with many: Carlos Areces, Rosella Gennari, Gabriel Infante López, Harald Hammarstrom, Juan Heguiabehere, Natasha Kurtonina, Christof Monz, and Hans de Nivelle.

References

[1] S. Abney. Partial parsing via finite-state cascades. *Natural Language Engineering*, 2:337–344, 1996.

[2] J. Allen. *Natural Language Understanding.* Benjamin Cummings, 1995.

[3] C. Areces. *Logic Engineering.* PhD thesis, ILLC, University of Amsterdam, 2000.

[4] C. Areces and M. de Rijke. From description to hybrid logic, and back. In *Advances in Modal Logic 3*, 2001.

[5] C. Areces, R. Gennari, J. Heguiabehere, and M. de Rijke. Tree-based heuristics in modal theorem proving. In W. Horn, editor, *Proceedings ECAI 2000*, 2000.

[6] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. In *Proceedings KR-92*, 1992.

[7] R. Baayen, R. Piepenbrock, and L. Gulikers. The CELEX lexical database (release 2). Distributed by the Linguistic Data Consortium, University of Pennsylvania, 1995.

[8] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press, 1999.

[9] R. Barzilay, K. McKeown, and M. Elhadad. Information fusion in the context of multi-document summarization. In *Proceedings of the*

37th Annual Meeting of the Association of Computational Linguistics (ACL'99), 1999.

[10] P. Blackburn and J. Bos. *Representation and Inference in Natural Language*. Studies in Logic, Language and Information. CSLI Publications, 2002.

[11] P. Blackburn, J. Bos, M. Kohlhase, and H. de Nivelle. Inference and computational semantics. In *Proceedings IWCS-3*, 1999.

[12] J. Bos. Doris 2001 system description. In *Proceedings ICoS-3*, 2001.

[13] C. Buckley, A. Singhal, and M. Mitra. New retrieval approaches using SMART: TREC 4. In D. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 25–48. NIST Special Publication 500-236, 1995.

[14] CLEF. URL: `http://www.clef-campaign.org`. Accessed August 6, 2001.

[15] Computing with Meaning. URL: `http://www.science.uva.nl/~mdr/Projects/ComputingwithMeaning/`. Accessed August 1, 2001.

[16] R. Cooper, D. Crouch, J. van Eijck, C. Fox, J. van Genabith, J. Jaspars, H. Kamp, D. Milward, M. Pinkal, M. Poesio, and S. Pulman. Using the framework. Deliverable D16, The FraCaS Consortium, 1996.

[17] **Derive**. URL: `http://www.science.uva.nl/~mdr/Projects/Derive/`. Accessed July 23, 2001.

[18] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Reasoning in description logics. In *Principles of Knowledge Representation*. CSLI Publications, 1996.

[19] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.

[20] K. Ferguson. *Measuring the Universe*. Headline, 1999.

[21] FraCaS: A Framework for Computational Semantics. URL: `http://www.cogsci.ed.ac.uk/~fracas/`. Accessed August 3, 2001.

[22] W. Frakes. Stemming algorithms. In W. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, pages 131–160. Prentice Hall, 1992.

[23] R. Gaizauskas. Evaluation in language and speech technology: introduction to the special issue. *Computer Speech and Language*, 12:249–262, 1998.

[24] C. Gardent and B. Webber. Automated reasoning and discourse interpretation. Claus report 113, Computational Linguistics, University of the Saarland, 2000.

[25] I. Gent, H. van Maaren, and T. Walsh, editors. *SAT 2000.* IOS Press, 2000.

[26] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures—the case study of modal K. In *Proceedings CADE-96*, 1996.

[27] H. Hammarstrom. Computing entailments in natural language texts. Master's thesis, Department of Computer Science, Uppsala University, 2001.

[28] S. Harabagiu, J. Burger, C. Gardie, V. Chaudri, R. Gaizauskas, D. Israel, C. Jacquemin, C.-Y. Lin, S. Maiorano, G. Miller, D. Moldovan, B. Ogden, J. Prager, E. Riloff, A. Singhal, R. Shrihari, T. Strzalkowski, E. Voorhees, and R. Weishedel. Issues, tasks, and program structures to roadmap research in question & answering (Q&A). URL: `http://www-nlpir.nist.gov/projects/duc/roadmapping.html`, October 2000.

[29] D. Harman. How effective is suffixing? *Journal of the American Society for Information Science*, 42:7–15, 1991.

[30] J. Heguiabehere and M. de Rijke. The random modal QBF test set. In *Proceedings IJCAR Workshop on Issues in the Design and Experimental Evaluation of Systems for Modal and Temporal Logics*, 2001.

[31] A. Heuerding and S. Schwendimann. A benchmark method for the propositional modal logics K, KT, and S4. Technical report IAM-96-015, University of Bern, Switzerland, 1996.

[32] L. Hirschman. The evolution of evaluation: lessons from the Message Understanding Conferences. *Computer Speech and Language*, 12:281–307, 1998.

[33] J. Hobbs et al. FASTUS: a cascaded finite-state transducer for extracting information from natural-language text. In E. Roche and Y. Schabes, editors, *Finite State Devices for Natural Language Processing.* Cambridge MA: MIT Press, 1996.

[34] I. Horrocks, P.F. Patel-Schneider, and R. Sebastiani. An analysis of empirical testing for modal decision procedures. *Logic Journal of the IGPL*, 8:293–323, 2000.

[35] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. In *Proceedings IJCAI-97*, pages 202–207, 1997.

[36] G. Infante and M. de Rijke. Playing with fragments. Manuscript, 2001.

[37] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Prentice Hall, 2000.

[38] A. Kilgariff and A. Palmer. Introduction to the Special Issue on SENSEVAL. *Computers and Humanities*, 34:1–13, 2000.

[39] A. Koller. Insert the zorkmid then kick the dispenser: computational linguistics and theorem proving in a computer game. In *Proceedings ICoS-3*, 2001.

[40] W. Kraaij and R. Pohlmann. Viewing stemming as recall enhancement. In *Proceedings SIGIR'96*, pages 40–48, 1996.

[41] W. Kraaij and R. Pohlmann. Comparing the effect of syntactic vs. statistical phrase index strategies for Dutch. In *Proceedings ECDL'98*, pages 605–617, 1998.

[42] N. Kurtonina and M. de Rijke. Expressiveness of concept expressions in first-order description logics. *Artificial Intelligence*, 107:303–330, 1999.

[43] R. Ladner. The computational complexity of provability in systems of modal logic. *SIAM Journal on Computing*, 6:467–480, 1977.

[44] I. Mani and E. Bloedorn. Summarizing similarities and differences among related documents. *Information Retrieval*, 1(1–2):35–67, 1999.

[45] F. Massacci. Design and results of the Tableaux-99 non-classical (modal) system competition. In *Proceedings Tableaux'99*, 1999.

[46] C. Monz and M. de Rijke. Light-weight entailment checking for computational semantics. In *Proceedings ICoS-3*, 2001.

[47] C. Monz and M. de Rijke. The University of Amsterdam at CLEF-2001. In C. Peters, editor, *Working Notes CLEF-2001*, 2001.

[48] T.S. Morton. Coreference for nlp applications. In *Proceedings ACL-2000*, 2000.

[49] I. Moulinier, J. McCulloh, and E. Lund. West Group at 2001: Non-English monolingual retrieval. In *Proceedings CLEF-2000*, 2000.

[50] MSPASS V 1.0.0t.1.2.a. URL: `http://www.cs.man.ac.uk/~schmidt/mspass`. Accessed February 23, 2001.

[51] H.-J. Ohlbach, A. Nonnengart, M. de Rijke, and D.M. Gabbay. Encoding two-valued non-classical logics in classical logic. In J. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier, 2001.

[52] M. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

[53] I. Pratt-Hartmann. The two-variable fragment of English. In *Proceedings ICoS-3*, 2001.

[54] J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System — Experiments in Automatic Document Processing*. Prentice Hall, 1971.

[55] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, 1994.

[56] W.M. Shaw Jr, R. Burgin, and P. Howell. Performance standards and evaluations in IR test collections: Cluster-based retrieval methods. *Information Processing & Management*, 33:1–14, 1997.

[57] ACL SIGSEM: Special Interest Group in Computational Semantics. URL: `http://www.sigsem.org`. Accessed on May 17, 2000.

[58] SIRIdUS: Specification, Interaction and Reconfiguration in Dialogue Understanding Systems. URL: `http://www.cam.sri.com/siridus/`. Accessed August 9, 2001.

[59] K. Sparck Jones. Automatic indexing. *Journal of Documentation*, 30:393–432, 1974.

[60] K. Sparck Jones and J. Galliers. *Evaluating Natural Language Processing Systems: An Analysis and Review.* Springer Verlag, 1996.

[61] SPASS Version 1.0.3. URL: `http://spass.mpi-sb.mpg.de/`. Accessed May 23, 2000.

[62] T. Strzalkowski. *Natural Language Information Retrieval.* Kluwer, 1999.

[63] J. Sukkarieh. Quasi-NL knowledge representation for structurally-based inference. In *Proceedings ICoS-3*, 2001.

[64] A. Tacchella. *SAT system description. In *Proceedings DL'99*, 1999.

[65] TANCS: Tableaux Non-Classical Systems Comparison. URL: `http://www.dis.uniroma1.it/~tancs`. Accessed on January 17, 2000.

[66] TREC: Text Retrieval Evaluation Conference. URL: `http://trec.nist.gov`. Accessed on July 21, 2001.

[67] E.M. Voorhees. Natural language processing and information retrieval. In *Proceedings of Second Summer School on Information Extraction*, LNAI. Springer-Verlag, 1999.

[68] WordNet. URL: `http://www.cogsci.princeton.edu/~wn`. Accessed May 17, 2001.