Robust Minimal Recursion Semantics

WORKING PAPER — please do not distribute without checking with the author

Ann Copestake
aac@cl.cam.ac.uk

January 2004 (updated DTD and other minor changes June 2006)

Abstract

Deep and shallow processing techniques have complementary strengths and weaknesses. There are various ways in which deep and shallow techniques can be combined in practical systems to take advantage of the strengths of each approach. In this paper, I argue for the advantages of using a semantic representation as the interface between various types of deep and shallow processing. I illustrate how with a flat semantics approach, shallow processing can be regarded as producing a semantics which is underspecified with respect to deeper processing, but fully compatible with it. I describe the notion of specificity which results and illustrate its connection with specificity in typed feature structures. I also show how semantic composition rules designed for deep formalisms can be adapted to shallow processing. Some practical experiments are described using the tag sequence grammar from the RASP tools and the LinGO English Resource Grammar.[1]

1 Introduction

WARNING: This document is intended to explain the RMRS formalism rather than the details of analyses of particular examples. The details of the RMRSs in this document may not be in line with the current grammars. Please see the examples in the semantic test suite instead.

There are many highly-developed deep processing systems, often associated with linguistically-motivated formalisms including Head-driven phrase structure grammar (HPSG), Lexical-functional grammar (LFG), Tree-adjoining grammar (TAG, XTAG, LTAG), Generalized phrase structure grammar (GPSG) and various types of categorial grammar (CG). Some broad-coverage deep grammars have been developed, which cover a large proportion of linguistic phenomena. These grammars can be used to produce a detailed logical semantic representation. Formally, the grammars are expressed declaratively and in principle can be used for generation as well as analysis. Although formalisms such as HPSG have been considered to be practically intractable, well-engineered systems now exist which are fast enough for practical use in applications such as email response (see e.g., Oepen et al, 2003).

There are, however, at least three classes of problem which limit the practical utility of systems based on deep processing of text (speech raises other issues which will not be discussed here). The first is that the number of analyses returned can be far too high for any realistic prospect of filtering by a post-processor. In general, techniques such as packing merely delay the problem, since few applications can work off the packed representation. Integration of statistical techniques for parse selection with deep processing is promising, but is far from being a solved problem, especially with HPSG. The second problem is robustness: deep systems assume a notion of grammaticality so that input which does not meet the criteria embodied in the grammar fails to parse. For edited text, parse failures are generally caused by lack of coverage — often missing lexical information. Constructing detailed lexical information manually is time-consuming and error-prone, while automatic or semi-automatic techniques run into problems because of data sparsity. For unedited text, minor typographic mistakes can cause complete parse failure. The third issue is speed: deep processing techniques are not fast enough to process large volumes of text, as is required by information retrieval, information extraction and question answering, for instance.

There are a range of techniques for processing which provide some linguistic structure at a shallower level than the deep processors. These include part-of-speech tagging and NP chunking. Such techniques are much faster than deep processing and also more robust. Because they involve a less detailed representation, they tend to involve a lower degree of ambiguity, but more importantly, they are generally developed with a fully integrated statistical metric for selection. However, they do not construct sufficient information to support conventional semantic representation.

Parsers which produce relatively simple dependency structures without a detailed lexicon are intermediate in depth (e.g., the RASP system Briscoe and Carroll, 2002). They are faster and more robust than deep parsers, though not as efficient as the shallower techniques, but there is generally no treatment of long-distance dependencies, and the structures produced contain insufficient information to recover scope information. They allow ungrammatical strings and the lack of lexical subcategorization information still precludes conventional semantic representation.

[1]This work was funded by the EU-project Deep Thought. It has benefited greatly from discussions with Ted Briscoe, Dan Flickinger, Alex Lascarides, Stephan Oepen and Simone Teufel and with colleagues working on the Deep Thought project as well as other researchers at Saarbrücken, Edinburgh and Cambridge where versions of it were presented. As usual, however, all errors are the responsibility of the author.

Deep processing is inherently slower than shallow processing, because it involves a larger search space. It is inherently less robust, because it assumes some notion of grammaticality and requires more detailed lexical information. Conversely, although it is possible to enhance shallow processing to overcome some of the disadvantages mentioned above, this requires more search space and additional knowledge sources. Even if the additional knowledge can be automatically acquired, robustness will decrease, because of data sparsity. The deep/shallow dimension is somewhat associated with the distinction between manually- and automatically- constructed resources, but there are manually constructed shallow processing systems and some automatically learned intermediate processors. For the purposes of this paper, the most important distinction between deep and shallow processing concerns the lexical requirements: in particular, deep processing requires detailed subcategorization information while shallow processing just needs some indication of possible parts-of-speech, plus information about irregular morphology.

Question answering is one application where there is a natural division of labour between deep and shallow processing. The requirement to process large volumes of answer text means that some shallow approach is required, at least for identification of likely answers. On the other hand, there are far fewer questions to process, the questions tend to be relatively short and precise analysis is important. So deep processing of questions is feasible and useful. In particular, questions can involve long-distance dependencies which will not be identified by shallow techniques.

Another application involves intelligent summarization. Key passages of text can be identified robustly using cue phrases. For instance, in scientific papers it is possible to identify sentences which describe the goal of the paper (Teufel, 1999). The vocabulary and syntactic structures used by authors to introduce the goal are relatively constrained and could potentially be deep parsed, although the actual text that describes the work performed is much less constrained and therefore difficult to deep parse with sufficient robustness.

Information extraction is also an application which should benefit from combining deep and shallow processing. An IE system needs to be able to process very large quantities of text, so it cannot be attempted by deep processing alone. On the other hand, most existing shallow processing systems rely on large amounts of hand-coding for each new domain, and are not very precise. The ideal situation would be to use shallow processing with limited domain-tuning to identify interesting sentences and then robustly invoke a deep processor with a domain-independent grammar to precisely extract the information which is of interest.

Deep processing may be sufficiently efficient for email answering. But lack of robustness causes problems, which could potentially be alleviated by using shallow processing as a backoff technique.

There are various strategies for combining deep and shallow processing (for specific approaches, see e.g., Crysmann et al, 2002; Daum et al, 2003). One involves augmenting a deep processor with shallow techniques. This may be appropriate in circumstances where deep processing is fast enough and the requirement is to increase robustness in the face of parse failure. For instance, a chart may be constructed via deep processing and, if full parsing fails, the substructures can be connected by more robust methods (e.g., Kasper et al, 1999). Another approach is to preprocess with a shallow parser and to use the information produced to guide the deeper parser. This is commonly done with POS tagging, which can be used as a preprocessor for deep parsing, but intermediate parsers could also be used in this way (if their output were compatible with the deeper processing). This cuts down ambiguity and thus increases speed, but only improves robustness if there is some strategy for backing-off to the shallow parse results if the deep parse fails. A less-explored strategy is to invoke shallow processing first and to deep process only selected parts of the input, as suggested above for IE.

All these strategies require integration of the deep and shallow processors, to differing degrees. Normally, integration at the syntactic level is attempted. For instance, a POS tagger's categories can be translated into the classes used by a deep parser in order to disambiguate. But even this case, which is the simplest we might consider, raises problems. For instance, a POS tagger may systematically tag *-ing* forms occurring in a prenominal positional as adjectives, while a deep parser might treat them as gerunds. Individual discrepancies like this do not pose a large problem, but each tagger/parser pair will require different adjustments and the cumulative effect is non-trivial. Much more serious issues arise with shallow parser that provide some bracketing of constituents, since the deep and shallow parsers may not agree on this, especially given that the underlying linguistic theory between different deep parsers gives rise to alternative bracketings. There seems little chance of agreement on a canonical syntactic structure. This is also illustrated by the difficulty of providing meaningful comparisons between parsers unless there is an agreed treebank. Carroll et al (1998) argue for comparison to be based on grammatical relations, but this also has its critics.

The alternative proposed here is to produce semantic representations from deep and shallow parsers, aiming for a maximal degree of semantic compatibility. The most important reason to take this approach is that work on providing a standardised semantic representation would be useful for many purposes: allowing different parsers (and generators) to be connected up to a variety of underlying systems. Standardising grammatical relations is not directly useful in this way. Furthermore, we can view semantic representations constructed from shallow parsers as underspecified forms of the output from a deep parser in a way that can be made formally precise. Finally, any constraints that arise from the domain are most naturally expressed in terms of semantics and it is potentially very useful to be able to apply domain constraints to both deep and shallow parsing.

The types of integration mentioned above are all possible under this strategy, on the assumption that semantic structures are produced compositionally. A deep parse failure involves disconnected semantic structures, which may be connected via shallow parsing, possibly with the help of domain constraints. Semantic structures constructed from a shallow parser can guide deep parsing, because only constituents with compatible semantics need be considered. Finally, in an application such as IE, the patterns necessary to identify regions of potential interest can be robustly and succinctly expressed in terms of semantics, and deep parsing can be used to further instantiate this information if shallow parsing is produces results which are too underspecified be be used by themselves.

To summarise, making shallow parsing return underspecified semantic representations has the following advantages:

- Integrated parsing: shallow parsed phrases could be incorporated into deep parsed structures.
- Deep parsing could be invoked incrementally in response to information needs.
- Reuse of knowledge sources is possible: information for further processing of shallow parsing might also be used on deep parser output. (e.g., domain knowledge, recognition of named entities, transfer rules in MT)
- The formal properties should be clearer, and thus the representations might be more generally usable.

In the remainder of this paper, I will describe a proposed semantic representation that can function as the interface between deep and shallow processing. In section 2 I will give an initial simple example and illustrate how shallow parsers can be used to give compatible output to a deep parser. The specific semantic representation introduced here is a modification of Minimal Recursion Semantics (MRS: Copestake et al 1999), known as Robust MRS (RMRS). Section 3 gives more details of the formal properties of RMRS. Section 4 briefly describes RASP and introduces RASP-generated RMRSs. In section 5, I describe semantic composition with RMRS using rules associated with the RASP tag-sequence grammar, and show that it is compatible with the algebra used for MRS. Section 7 discusses some initial approaches for using RMRS, and finally section 8 contains conclusions and further work.

2 A simple example of RMRS

In this section, RMRS is introduced by example. I will consider the semantic representations for the sentence:

(1) every fat cat sat on some table

The scoped semantic representations for this sentence can be assumed to be the following:

$$\text{every}(x, \text{fat}(x) \wedge \text{cat1}(x), \text{some}(y, \text{table1}(y), \text{sit1}(e_{spast}, x) \wedge \text{on2}(e', e, y)))$$

$$\text{some}(y, \text{table1}(y), \text{every}(x, \text{fat}(x) \wedge \text{cat1}(x), \text{sit1}(e_{spast}, x) \wedge \text{on2}(e', e, y)))$$

In fact, this is a slightly simplified form of the scoped semantic representation output by the April 2003 version of the LinGO English Resource Grammar (ERG: Copestake and Flickinger, 2000).[2] Here, I will assume that it is relatively uncontroversial. The representation uses generalized quantifiers — this is a fundamental assumption in what follows. It also assumes a neo-Davidsonian approach using event variables, and a simple account of tense as a predicate on events: these assumptions are less critical to the general RMRS idea. The assumption of the 'extra' event in prepositions is motivated by predicative use after a copula: this follows the current treatment in the ERG, but is not important for RMRS.

The central idea in this paper is that this fairly conventional semantic representation can be directly related to very underspecified semantic information. To take the extreme case, I will argue that a POS tagger can be viewed as providing very underspecified semantic representation, and then show how this is formally related to the structures shown above.

2.1 Semantics from a POS tagger

A conventional POS-tagger output for the example sentence is as follows:

(2) every_AT1 fat_JJ cat_NN1 sat_VVD on_II some_DD table_NN1

[2]As will be discussed later, the current version of the ERG now adopts a much more standardised naming convention for predicates.

The POS tagger can be viewed as providing some semantic information:

- The lemmatization identifies a class of relations. For instance, in the example above, *sat* is lemmatized to *sit*. On the usual assumption of a convention whereby relation names (i.e., predicates) are related to the spelling of the lemma, we can associate the tagged word with a name corresponding to the class of all relations that correspond to that spelling.
- The POS tag provides partial disambiguation of the class of relations corresponding to the lemma's spelling: e.g., we can distinguish the noun *table* from the verb.
- The POS tag for inflected forms can be used to identify the semantic contribution of the inflection.

However POS tagging provides no relational information and no information about scope. Intuitively, therefore, the semantic information content is very underspecified compared to the semantics from the deep processor.

The structure below is an initial proposal for the representation of the semantic information associated with the POS tagged sentence:

$$\text{\_every\_q}(x0), \text{\_fat\_j}(x1), \text{\_cat\_n}(x2), \text{\_sit\_v}(e3_{past}), \text{\_on\_p}(e4), \text{\_some\_q}(x5), \text{\_table\_n}(x6)$$

For now, this should be taken informally, as an indication of what the logical representation could contain. Details will be discussed later on. Note, however, that the representation identifies some entities which are involved, but essentially constructs one entity per word in the input: for instance, x0 and x1 are not related. The variables associated with the entities are sorted, so that 'x's are object-denoting and 'e's are event-denoting. Tense information is assumed to be encoded via subsorts of event.

Since there is no relational information, the predicates have to be unary. Notice also that the naming convention for the relations allows them to be generated automatically. The leading underscore identifies lexical predicates: this will be made clearer later on. Predicates all have their broad part of speech indicated based on the tag: semantically this corresponds to a very broad sense distinction.

The crucial assumption is that this semantics can be generated from a 'lexicon' for the tags, rather than a genuine lexicon which is indexed by lexeme. Specifically, the tag lexicon assumed here is as follows:

AT1	lexrel(x)
JJ	lexrel_j(x)
NN1	lexrel_n(x)
VVD	lexrel_v(e_{past})
II	lexrel_p(e)
DD	lexrel_q(x)

Here, 'lexrel' is a special symbol, which is taken to be replaced by the individual lemma (with a leading underscore). Producing the semantics from the tagger output and this lexicon is a trivial matter. All variables are taken to be different unless repeated in the same lexical entry. All tags correspond to one or more relation/argument combination (a relation plus its arguments will be referred to as an **elementary predication**). The elementary predications are simply concatenated to give the semantic representation for the sentence, hence no sort of tree structure is needed to guide composition.

At this point, there are two main questions:

- Can the proposal that this is an underspecified semantic representation be made formally precise?
- How do we manipulate syntax of the semantic representation which is generated by deep processing in order to make the connection with the shallow semantics more obvious?

The answer to the first question follows established approaches to underspecification, in particular, Copestake et al (1999) and Copestake et al (2001). The shallow semantics is interpreted in terms of the full representations it can be expanded to. In what follows, I will go through the main steps required to answer the second question. The central idea is to split up the deep representation into minimal units which can be manipulated independently. The shallow semantics then contains only some of these units.

2.2 Flat semantics, underspecified scope (MRS)

The first step is to adopt a version of the deep semantics which is capable of underspecifying scope, and in which the elementary predications are flattened, so that the essential rule of composition is that elementary predications are appended. The approach I will assume is Minimal Recursion Semantics (MRS, Copestake et al, 1999), although Bos' (1995) semantics would be another candidate.

For convenience, the scoped structures for our sample sentence are repeated below:

$$\text{every}(x, \text{fat}(x) \wedge \text{cat1}(x), \text{some}(y, \text{table1}(y), \text{sit1}(e_{spast}, x) \wedge \text{on2}(e', e, y)))$$

$$\text{some}(y, \text{table1}(y), \text{every}(x, \text{fat}(x) \wedge \text{cat1}(x), \text{sit1}(e_{spast}, x) \wedge \text{on2}(e', e, y)))$$

The single MRS structure corresponding to these scopes is as follows:

$l0$:every$(x, h1, h2)$, $l1$:fat(x), $l1$:cat1(x), $l4$:sit1(e_{spast}, x), $l4$:on2(e', e, y), $l5$:some$(y, h6, h7)$, $l6$:table1(y), qeq$(h1, l1)$, qeq$(h6, l6)$

I will not go through details of the MRS structure here, see Copestake et al (1999) for details. The salient points are:

- The MRS can be specialised to give exactly the two scoped structures.
- Each elementary predication is labelled $l0$, $l1$ etc.
- Scopal arguments in the scoped representation are indicated by holes: $h1$, $h2$ etc. Holes may be filled by labels according to the constraints on scope. In the case of the example above, the two possible scopes correspond to $h1 = l1, h2 = l5, h6 = l6, h7 = l4$ and $h1 = l4, h2 = l5, h6 = l6, h7 = l0$.
- Constraints on scope are given above by qeq conditions, but since the details of these are unimportant for the discussion in the current work, they will not be further explained here.

In standard MRS, implicit conjunction is indicated by equality between labels. For instance, l1 labels both $l1$:fat(x) and $l1$:cat1(x). However, this is not essential, and in what follows, I use a variant where labels on elementary predications are unique and conjunction arising from intersective modification is explicitly represented. This corresponds to:

$l0$:every$(x, h1, h2)$, $l1$:fat(x), $l2$:cat1(x), $l4$:sit1(e_{spast}, x), $l14$:on2(e', e, y),
$l5$:some$(y, h6, h7)$, $l6$:table1(y),
qeq$(h1, l3)$, qeq$(h6, l6)$, in-g$(l2, l1)$, in-g$(l4, l14)$

Here 'in-g' stands for 'in group': this representation allows for the fact that there may be an indefinitely large number of elements in a conjunction (see §2.9, below).

MRS takes us quite a long way towards the goal of splitting the semantics into individual units, but it still assumes that we have knowledge of the arity of the predicates, which does not apply to the shallow semantics situation. To allow for this, we need to move to a representation where arguments can be considered individually.

2.3 Parsons style representation

The following shows a more factorised representation where the base predicates are all maximally unary and the arguments are represented by separate (binary) relations:

$l0$:every(x), $l1$:fat(x), $l2$:cat1(x), $l4$:sit1(e_{spast}), $l14$:on2(e'), $l5$:some(y), $l6$:table1(y),
qeq$(h1, l1)$, qeq$(h6, l6)$, in-g$(l2, l1)$, in-g$(l4, l14)$
RSTR$(l0, h1)$, BODY$(l0, h2)$, RSTR$(l5, h6)$, BODY$(l5, h7)$,
ARG1$(l4, x)$, ARG1$(l14, e)$, ARG2$(l14, y)$

Conversion from MRS to this representation merely involves some naming convention for the arguments of each type of predicate. The default naming convention is ARG1, ARG2 etc, with some relations corresponding to closed class words, such as determiners, having more characteristic argument names (e.g. RSTR and BODY for quantifiers). In this respect, the notation is close to the typed feature structures used internally to represent semantics in HPSGs such as the ERG.

I refer to this as 'Parsons style' notation, since it relates to Parsons' (1990) proposal for making arguments first-class predications. For instance, Parsons argued that kick$(e) \wedge$ agent$(e, x) \wedge$ patient(e, y) is a preferable notation to kick(e, x, y). Parsons reasons for advocating this approach are somewhat irrelevant to our concerns in this paper, but the reason for using this style of notation here is directly related to Dowty's (1989) discussion of Parsons' approach. Dowty observed that this style of representation is better suited to nominalizations than the conventional logical form because nominals' arguments are generally optional, and this notation makes it easier to omit arguments. The lack of lexicon in shallow semantics means we can't know the arity of open-class predicate, but Parsons' style of representation allows incremental addition of arguments. This amounts to treating them as optional in terms of the syntax of the semantics.

There are two significant differences from Parsons. The first is the use of ARG1 etc rather than agent/patient. The primary reason for this is to allow a simple scheme which does not require a lexicon: it is possible to map it into a more detailed scheme (this is discussed in Copestake and Flickinger, 2003). As I will discuss in §3.1, underspecification of ARG-relations allows for dependency structures where we know there is some relation but can't be sure which one it is.

The second distinction is the use of labels instead of events as the anchor for the arguments. This allows the same approach to be used for cases where events are not motivated such as relational nouns (e.g., *group*) and scopal adverbs and connectives (e.g., *not*, *probably*, *but*).

This notation requires that individual elementary predications have distinct labels. This is why the implicit representation of conjunction in 'standard' MRS is not suitable and the alternative representation was adopted above.

2.4 Distinct variable names, plus equalities

The separation of arguments is the major difference between the proposed representation and MRS, but there are two further steps. The first is to have distinct names for variables and to represent identity between them with equalities. The reason for this is to make it clear what extra information is involved: the information that variables are equated can be added incrementally.

$l0$:every$(x0)$, $l1$:fat$(x1)$, $l2$:cat1$(x2)$, $l4$:sit1$(e3_{spast})$, $l14$:on2$(e4)$
$l5$:some$(x5)$, $l6$:table1$(x6)$,
qeq$(h1, l1)$, qeq$(h6, l6)$, in-g$(l2, l1)$, in-g$(l4, l14)$
RSTR$(l0, h1)$, BODY$(l0, h2)$, RSTR$(l5, h6)$, BODY$(l5, h7)$ARG1$(l4, x2)$, ARG1$(l14, e3)$, ARG2$(l14, x5)$,
$x0 = x1, x1 = x2, x5 = x6$

2.5 A predicate naming convention

Finally, a predicate naming convention is required so that the automatically constructed predicates in the tagger output can be straightforwardly related to the deep representation. This is indicated below:

$l0$:_every_q$(x0)$, $l1$:_fat_j$(x1)$, $l10$:_cat_n$(x2)$, $l4$:_sit_v_1$(e3)$, $l14$:on$(e4)$,
$l5$:some$(x5)$, $l6$:table_N_1$(x6)$,
qeq$(h1, l1)$, qeq$(h6, l6)$, in-g$(l2, l1)$, in-g$(l4, l14)$
RSTR$(l0, h1)$, BODY$(l0, h2)$, RSTR$(l5, h6)$, BODY$(l5, h7)$, ARG1$(l4, x2)$, ARG1$(l14, e3)$, ARG2$(l14, x5)$,
$x0 = x1, x1 = x2, x5 = x6$

Here sit_V_1 is more specific than sit_V, spast more specific than past, and table_N_1 more specific than table_N. This is discussed further in §3.1.

2.6 Robust underspecification

We have now arrived at a point where it can easily be seen that the tagger output is an underspecified form of the deep representation. The tagger output shown above was:

_every_q$(x0)$, _fat_j$(x1)$, _cat_n$(x2)$, _sit_v$(e3_{past})$, _on_p$(e4)$, _some_q$(x5)$, _table_n$(x6)$

The version with labels is:

$l0$:_every_q$(x0)$, $l1$:_fat_j$(x1)$, $l10$:_cat_n$(x2)$, $l4$:_sit_v$(e3_{past})$, $l14$:_on_p$(e4)$, $l5$:_some_q$(x5)$, $l6$:_table_n$(x6)$

This can be converted into the deep representation shown above by adding 'in-g's, 'qeq's, ARGn relations and equalities and by specialising predicates. To make the relationship clearer, I have used the same variable names in both cases, but in general, alphabetic variance has to be taken into account.

Notice that the output from an NP chunker is intermediate in specificity, for instance:

$l0$:_every_q$(x0)$, $l1$:_fat_j$(x1)$, $l10$:_cat_n$(x2)$, $l4$:_sit_v$(e3_{past})$, $l14$:_on_p$(e4)$, $l5$:_some_q$(x5)$, $l6$:_table_n$(x6)$
in-g$(l10, l1)$, $x0 = x1, x1 = x2$

The extraction of semantic representations which are of intermediate specificity is discussed in some detail in §4 with reference to the RASP system.

2.7 Lexical requirements for RMRS

In general terms, the lexical requirements for construction of RMRSs are as follows:

1. A tag lexicon or similar class-based lexicon is needed for shallow processing.
2. Consistent naming conventions are required for relations corresponding to open-class words, depending only on the orthography of the lemma and on the POS tag.
3. The representations produced by deeper analysis be in a well-defined relationship to shallower predicates if they make finer-grained distinctions:
   e.g., _sit_v_1 must be taken to be more specific than _sit_V
4. There is a consistent convention for ARG1 etc which depends on syntax.
5. A lexicon is required for some closed class words for shallow/intermediate processing (e.g., we don't want any semantics for some closed class words such as *do*). On the assumption that the semantics from the deep analsys is taken as normative, this forms part of the meta-level semantic interface (SEM-I) which is discussed in Copestake and Flickinger (2003).
6. No lexicon is required for open-class words.

2.8 Characterisation

The problem of comparing two RMRS structures is potentially exponential. In the particular case where the structures have been built from the same string, indexing can be used to allow for efficient comparison. We index by characters, with the origin of the elementary predications being shown by a character range, and call this **characterisation**. For instance, consider the example input sentence:

(3) Every cat sat on some table.

The characters can be indexed as follows:

e	v	e	r	y		f	a	t		c	a	t		s	a	t
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	o	n		s	o	m	e		t	a	b	l	e	.		
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		

Indexing the elementary predications leads to the following tagger output, where the indices are shown before the elementary predications:

$$\langle 0,4\rangle{:}l0{:}\_\text{every\_q}(x0),\ \langle 6,8\rangle{:}l1{:}\_\text{fat\_j}(x1),\ \langle 10,12\rangle{:}l10{:}\_\text{cat\_n}(x2),\ \langle 14,16\rangle{:}l4{:}\_\text{sit\_v}(e3_{past}),$$
$$\langle 18,19\rangle{:}l14{:}\_\text{on\_p}(e4),\ \langle 21,24\rangle{:}l5{:}\_\text{some\_q}(x5),\ \langle 26,30\rangle{:}l6{:}\_\text{table\_n}(x6)$$

The reason for using character positions rather than word positions to index is that we cannot assume that two systems will tokenize the input in exactly the same way. The character position indices must be assigned on the basis of the raw input, before any sort of preprocessing has been carried out.

A complication arises when elementary predications are introduced by a construction. In this case, they are indexed by the initial and final character string of the elements of the construction. Since qeqs, equalities etc relate variables which are introduced by elementary predications, there should be no need to index these. The variable equivalences across two RMRSs are established by the equivalences between elementary predications.

2.9 Conjunction and the use of in-g

In the verson of MRS used in the ERG, sharing of labels corresponds to conjunction. This is used in the representation of ordinary adjectives and adverbs, prepositional modifiers and (some) PP arguments. The point is that conjunction arising from such situations is ubiquitous and explicitly representing it clutters the representation unnecessarily. Furthermore, the label sharing device allows a canonical representation.

In RMRS, the labels cannot be used in this way, because they have to act as the locus for attachment of arguments. Thus labels on individual EPs are distinct. To represent implicit conjunction, the IN-G ('in group') notation is used. When two labels are connected by an IN-G relationship, this indicates conjunction: i.e, it corresponds to the coindexation of labels of EPs in MRS.

It is convenient to have a notion of a canonical set of IN-Gs This can be regarded in the same way as the set of qeqs - although different groups of qeqs might semantically result in the same thing, if we assume consistent composition principles, we'll get the same set of qeqs for comparable strings. The same is true of IN-Gs - although semantically they just correspond to conjunction, we can compare much more efficiently if we assume rules about how they are constructed. This also allow us to underspecify some attachments and possibly change the interpretation of IN-Gs, but this is ignored for now.

The assumption is that whenever an IN-G is created, it corresponds to a situation where one label can be considered 'higher'. Higher could be defined in several ways but we assume that the modifiee is always 'higher' (this makes sense because it may have several modifiers) and that a semantic head is always higher otherwise.

This is complicated in the context of conversion from the ERG because we don't know what the composition rules are, so we have to try and simulate on the basis of which label comes from a more 'major' relation.

3 An outline of the approach to specificity

Specificity is defined in terms of the syntax of the semantic representation, not the denotation (cf MRS algebra semantics, Copestake et al, 2001, where the models are fully specified LFs).

Informal definition: a semantic representation A is an underspecified form of B if A can be converted to B by:

1. adding argument-relations (e.g., ARG1)

2. adding equalities between variables

3. specialising predicates

The formal properties depend on treating representations as made up of minimal units of semantic information. Specificity is then treated in terms of a partial order on structures in terms of information containment (cf (typed) feature structures).

3.1 Underspecification of relations

The partial order on individual elementary predications is determined by a partial order on predicates. For instance: _sit_v_1 $\sqsubset$ _sit_v

There is also a partial order on the argument relations, so that ARG1 $\sqsubset$ ARGN etc. This allows for the situation where an analyser can determine that some argument position is filled by a variable, but the specific position cannot be determined. A refinement of this is to use a hierarchy of relations:
ARG2 $\sqsubset$ ARG2-3, ARG3 $\sqsubset$ ARG2-3, ARG2-3 $\sqsubset$ ARGN etc.

For further details of the relationship to thematic roles, see Copestake and Flickinger (2003).

Finally, variable sorts are in a hierarchy, e.g., spast $\sqsubset$ past.

3.2 MRS to RMRS interconversion

The interconversion of MRSs produced by the ERG and RMRSs is largely straightforward. The main steps are as follows:

1. Convert all variable sorts to the standardised RMRS set.

2. Convert the MRS implicit conjunction to the explicit 'in-group' representation.

3. Split off argument relations, ignoring any uninstantiated optional arguments.

The reverse process is possible if the RMRS is sufficiently instantiated. However, we intend to make the generator run directly off RMRSs, so the main need to construct MRSs is for situations where completeness needs to be guaranteed — for instance, conversion to a fully scoped representation.

4 Intermediate level processing

In this section, I consider constructing RMRSs from the RASP system (Robust accurate domain-independent statistical parsing: Briscoe and Carroll, 2002). RASP consists of a tokenizer, POS tagger, lemmatizer, tag-sequence grammar, statistical disambiguator. The central part of RASP from our current perspective is the grammar, which is a detailed grammar for (written) English, written in a constraint-based formalism. RASP is robust because the lexicon only contains POS tags. For instance:

```
WORD VVD : V0[FIN +, AUX -, VFORM PAST, CONJ -].
```

The rules mention SUBCAT, but this is not lexically instantiated:

```
PSRULE S/np_vp : V2[WH -, INV -] --> N2 H1[FIN +].
PSRULE V/np : V1 --> H0[VSUBCAT NP] N2.
```

The system is in fact used for experiments on acquisition of subcategorization information. It can be regarded as being of intermediate depth: it is quite robust, but uses a weak notion of grammaticality and has minimal lexical requirements. It supports a variety of output formats, including grammatical relations and parse trees. The current work extends this by allowing RMRS output.

4.1 Robust semantics from RASP

WARNING: the details in this section are out of date. Please see Ritchie (2004) for more detailed discussion.

The current system for constructing RMRSs from RASP works off the tree output, keyed by the grammar rules. For instance:

```
(|T/txt-sc1/----|
 (|S/np_vp| (|NP/det_n1| |Every:1_AT1| (|N1/n| |cat:2_NN1|))
  (|V1/v| |bark+ed:3_VVD|)))
```

gives rise to the following RMRS:
$h24$:prpstn_m_rel$(h26), h1$:_every_q_rel$(x2), h3$:_cat_n_rel$(x2), h10$:_bark_v_rel$(e11)$,
ARG1$(h10, x2)$, RSTR$(h1, h8)$, BODY$(h1, h5)$, qeq$(h14, h10)$, qeq$(h8, h3)$
This is actually output in XML format as (partially) shown below:[3]

```
<rmrs>
<label vid='12'/>
<ep><gpred>prpstn_m_rel</gpred><label vid='12'/><var sort='h' vid='14'/></ep>
<ep><realpred lemma='every' pos='q'/><label vid='1'/><var sort='x' vid='2'/></ep>
<ep><realpred lemma='cat' pos='n'/><label vid='3'/><var sort='x' vid='2'/></ep>
<ep><realpred lemma='bark' pos='v'/><label vid='10'/><var sort='e' vid='11'/></ep>
<rarg><rargname>ARG1</rargname><label vid='10'/><var sort='x' vid='2'/></rarg>
<rarg><rargname>RSTR</rargname><label vid='1'/><var sort='h' vid='8'/></rarg>
<rarg><rargname>BODY</rargname><label vid='1'/><var sort='h' vid='5'/></rarg>
<hcons hreln='qeq'><hi><var sort='h' vid='14'/></hi><lo><label vid='10'/></lo></hcons>
<hcons hreln='qeq'><hi><var sort='h' vid='8'/></hi><lo><label vid='3'/></lo></hcons>
</rmrs>
```

The tag 'lexicon' that is used in the composition is the following:

```
<le>
<tag>AT1</tag>
<semstruct>
<hook><index>X</index><label>H</label></hook>
<ep><pred></pred><label>H</label><arg>X</arg></ep>
</semstruct>
</le>

<le>
<tag>NN1</tag>
<semstruct>
<hook><index>X</index><label>H</label></hook>
<ep><pred></pred><label>H</label><arg>X</arg></ep>
</semstruct>
</le>
```

[3]The DTD is given in the appendix. In this example, variable sorts are omitted, as are character positions. Variable identity is used instead of explicit equalities.

```
<le>
<tag>VVD</tag>
<semstruct>
<hook><index>E</index><label>H</label></hook>
<ep><pred></pred><label>H</label><arg>E</arg></ep>
</semstruct>
</le>
```

The lexical entries identify a ‘hook’ (see next section) and one or more elementary predications.

The grammar rules which are keyed off the tree are:

```
<rule>
<name>T/txt-sc1</name>
<dtrs><dtr>S</dtr><dtr>OPT</dtr><dtr>OPT</dtr><dtr>OPT</dtr>
                                              <dtr>OPT</dtr></dtrs>
<head>S</head>
</rule>

<rule>
<name>S/np_vp</name>
<dtrs><dtr>NP</dtr><dtr>VP</dtr></dtrs>
<head>RULE</head>
<semstruct>
<hook><index>E</index><label>H1</label></hook>
<ep><gpred>PRPSTN_M_REL</gpred><label>H1</label><var>H2</var></ep>
<rarg><rargname>ARG1</rargname><label>H0</label><var>X</var></rarg>
<hcons hreln='qeq'><hi><var>H2</var></hi><lo><var>H</var></lo></hcons>
</semstruct>
<equalities><rv>X</rv><dh><dtr>NP</dtr><he>INDEX</he></dh></equalities>
<equalities><rv>H</rv><dh><dtr>VP</dtr><he>LABEL</he></dh></equalities>
<equalities><rv>H0</rv><dh><dtr>VP</dtr><he>ANCHOR</he></dh></equalities>
<equalities><rv>E</rv><dh><dtr>VP</dtr><he>INDEX</he></dh></equalities>
</rule>

<rule>
<name>NP/det_n1</name>
<dtrs><dtr>DET</dtr><dtr>N</dtr></dtrs>
<head>RULE</head>
<semstruct>
<hook><index>X</index><label>H2</label></hook>
<rarg><rargname>RSTR</rargname><label>H</label><var>H1</var></rarg>
<rarg><rargname>BODY</rargname><label>H</label><var>H2</var></rarg>
<hcons hreln='qeq'><hi><var>H1</var></hi><lo><var>H3</var></lo></hcons>
</semstruct>
<equalities><rv>X</rv><dh><dtr>DET</dtr><he>INDEX</he></dh></equalities>
<equalities><rv>H</rv><dh><dtr>DET</dtr><he>LABEL</he></dh></equalities>
<equalities><rv>H3</rv><dh><dtr>N</dtr><he>LABEL</he></dh></equalities>
<equalities><dh><dtr>N</dtr><he>INDEX</he></dh>
<dh><dtr>DET</dtr><he>INDEX</he></dh></equalities>
</rule>

<rule>
<name>N1/n</name>
<dtrs><dtr>N</dtr></dtrs>
<head>N</head>
</rule>

<rule>
```

```
<name>V1/v</name>
<dtrs><dtr>V</dtr></dtrs>
<head>V</head>
</rule>
```

Grammar rules identify the semantic head daughter (which is not necessarily identical to the syntactic head). They may also specify argument relations and elementary predications. The formalism for RMRS composition in general is discussed in more detail in the next section.

4.2 Limitations on deep/shallow compatibility

Our approach to developing the semantic composition rules for RASP depends on the assumption that the deep grammar is to be taken as normative, but that the RASP system should not require access to any lexical material for open-class words. This approach breaks down in some cases. See Ritchie (2004).

5 An outline of RMRS composition

Semantic composition needs to be formalised for both deep and shallow systems. Since we are interested in investigating tight integration of deep and shallow processing, where we may need to combine phrases produced by a deep processor with those from a shallower system, the issue of composition is very important. In particular, compatible composition rules ensure that this sort of integration is feasible.

In this section, I briefly introduce the MRS composition principles described in Copestake et al (2001) and go on to show their relevance to composition with a POS tagger (trivially) and RASP.

5.1 A semantic algebra

Copestake et al (2001) define a semantic algebra which is intended to capture 'best practice' in semantic composition in an HPSG using MRS. Although semantic composition is implemented in terms of operations on typed feature structures, it is desirable to restrict the way that semantic structures are manipulated, both as a way of capturing linguistic generalizations about composition and also as an aid to designing efficient algorithms. The algebra is illustrated below with a simple example (ignoring scope). Semantic structures consist of four basic parts:

hook Hooks are used to give arguments for relations. They can be thought of as pointers into the relations which are set up lexically: they are the only way of accessing parts of the semantics of a sign. Hooks are functionally similar to lambda variables, although formally quite different. In a full system, hooks consist of three parts: an index, a label (the local top in MRS terms, see Copestake et al, 1999), and an external argument.

slots Slots are structures corresponding to syntactic unsaturation which also specify how the semantics is combined. A syntax 'slot' in one sign will be instantiated by being equated with a hook in another sign.

rels The bag of elementary predications. This is always accumulated in composition, though some rules may add relations themselves.

equalities The results of equating slots and hooks.

For scoped structures, qeqs are also required, but these are not important here.

Composition always involves the following principles:

1. the rels of the phrase are equal to the append of the the rels of the daughters plus any rels contributed by the construction
2. the equalities of the phrase are equal to the append of the the equalities of the daughters plus any equalities contributed by the construction
3. a (syntactically specified) slot in one structure (the daughter which corresponds to the **semantic head**) is filled by the hook of the other structure (by adding equalities)
4. the hook of the phrase is the semantic head's hook
5. the slots of the phrase are the unfilled slots of the daughters

The access to the semantics of a sign during composition is only via the hook and the slots.

The following should serve to illustrate the algebra (ignoring scope, qeqs etc):

	$hook$	$slots$	$rels$	$equalities$
cat :	$[x_2]$	$\{\}$	$[\text{cat}(x_2)]$	$\{\}$
every :	$[x_3]$	$\{[x_3]_{spec}\}$	$[\text{every}(x_3)]$	$\{\}$
every cat :	$[x_3]$	$\{\}$	$[\text{every}(x_3), \text{cat}(x_2)]$	$\{x_3 = x_2\}$

Here, the hook of *cat* has instantiated the spec slot of *every*, which is the semantic head (though not the syntactic head in the ERG). This leads to the equality between the variables in the result. The relations are accumulated. Since the slot is saturated, it is not carried on to the phrase.

The hook of *every cat* can then instantiate the subject slot of *sleep*, as below:

every cat :	$[x_3]$	$\{\}$	$[\text{every}(x_3), \text{cat}(x_2)]$	$\{x_3 = x_2\}$
sleeps :	$[e_1]$	$\{[x_1]_{subj}\}$	$[\text{sleep}(e_1, x_1)]$	$\{\}$
every cat sleeps :	$[e_1]$	$\{\}$	$[\text{sleep}(e_1, x_1), \text{every}(x_3), \text{cat}(x_2)]$	$\{x_1 = x_2, x_3 = x_2\}$

5.2 Semantic composition with a POS tagger

The semantic composition rules apply trivially in the case of POS-tagged text. In this situation, we have no knowledge of any syntax above the POS tag level, so there are no slots and the hooks are irrelevant. The composition principle of accumulation of elementary predications holds, however, so the semantics of the result involves an accumulation of the rels as specified in principle 1 above.

5.3 Simple semantic composition with RASP

The semantic composition mechanism in RASP obeys the principles of composition given above. It differs from the mechanism for composition in MRS in the ERG in some ways:

- Since there is no lexical subcategorization information in RASP, slots are always specified by rules rather than lexically. However, it is necessary for the label of the argument-taking structure to be visible at all points where arguments may be attached. For instance, in the example above, the label of the verb *bark* had to be visible at the point where the ARG1 relation was introduced. Although generally the label required will correspond to the label of the semantic head of some daughter, this is not the case if there is a scopal modifier (consider *every cat did not bark*). Access to the original label is done via an 'anchor', which is equated with the semantic head label in most cases, but which scopal modifiers set to be the anchor of the structure that they are modifying.

- Rules have their own semantic structures.

- The algebra formalisation requires binary combination: in order to model this, we can assume that rules are combined with daughters one-by-one. However, in practice, the combination of a single rule with all the daughters can be carried out in a single step. Generally one of the daughters is the semantic head, but sometimes the rule itself is, in effect, the semantic head.

- If no semantic rule is specified corresponding to a rule used in a tree, the rels are appended. If there is more than one daughter, the hook of the result contains new variables, since any of the daughters could correspond to the semantic head. Semantic composition is thus robust to omissions in the semantic component of the grammar: if no semantics is associated with any rules in the tree, the result is equivalent to a POS-tagged RMRS.

Despite these differences, the semantic structures that are output from phrases in RASP are, in principle, compatible with those that would be output by the ERG.

The (simplifed) structures below should illustrate this. The slots are indexed by the daughter names in the rule. The hooks and slots are shown with labels included, since this is needed for the ARG1 relation:

	$hook$	$slots$	$rels$	$equalities$
every cat :	$[l_8, x_3]$	$\{\}$	$[\text{every}(x_3), \text{cat}(x_2)]$	$\{x_3 = x_2\}$
sleeps :	$[l_9, e_1]$	$\{\}$	$[\text{sleep}(e_1)]$	$\{\}$
S/np_vp :	$[l_3, e_4]$	$\{[l_9, x_8]_{NP}, [l_3, e_4]_{VP}\}$	$[\text{ARG1}(l_3, x_2)]\{\}$	
	$[l_3, e_4]$	$\{\}$	$[\text{sleep}(e_1), \text{every}(x_3), \text{cat}(x_2), \text{ARG1}(l_3, x_2)]$	$\{x_3 = x_2\}$

6 Refinements

6.1 Inequalities

One extension to the RMRS formalism discussed above is a mechanism for specifying inequalities. To see why this is desirable, consider the following example:

(4) the donkey kicked the cat

This might receive the following semantics (donkey is the ARG1 of *kick*, while cat is the ARG2):

(5) $l0$:_the_q$(x1), l1$:_donkey_n$(x1), l2$:_kick_v$(e2), l3$:_the_q$(x3), l4$:_cat_n$(x3)$, ARG1$(l2, x1)$, ARG2$(l2, x3)$

We don't want to be able to further specialise to:

(6) $l0$:_the_q$(x1), l1$:_donkey_n$(x1), l2$:_kick_v$(e2), l3$:_the_q$(x3), l4$:_cat_n$(x3)$, ARG1$(l2, x1)$, ARG2$(l2, x3), x1 = x3$

This corresponds to the donkey and cat being identified.

Furthermore, 5 shouldn't be compatible with:

(7) $l0$:_the_q$(x1), l1$:_donkey_n$(x1), l2$:_kick_v$(e2), l3$:_the_q$(x3), l4$:_cat_n$(x3)$, ARG1$(l2, x3)$

where the cat is the ARG1.

Formally, the solution to this problem is to introduce inequalities:[4]

(8) the donkey kicked the cat

(9) $l0$:_the_q$(x1), l1$:_donkey_n$(x1), l2$:_kick_v$(e2), l3$:_the_q$(x3), l4$:_cat_n$(x3)$, ARG1$(l2, x1)$, ARG2$(l2, x3), x1 \neq x3$

It is important to note that we are not talking about the real world identity of entities, but rather about the models to which a given RMRS can be specialised. Hence it is irrelevant whether the utterance is about entities which are necessarily distinct or not.

Incorporating inequalities fully would lead to an extremely verbose representation and specifying them explicitly in grammar rules would be tedious at best. Luckily this is not required (at least currently), because in MRS as encoded in the ERG we have adopted an approach where the variables introduced by nouns are necessarily distinct from one another, and the same holds for verbs. i.e., for any two nominal relations, R1_n and R2_n, we can guarantee that if R1_n(x1) and R2_n(x2), then x1 $\neq$ x2, according to the ERG. This effect holds because constructions such as appositives, which might be supposed to equate nominal variables, actually introduce their own relations.[5]

One interesting alternative possibility that has not been investigated yet is to assume inequalities by default and allow equalities to override them. The formalisation of RMRS in terms of partial order of information content should be essentially compatible with the version of defaults described by Lascarides and Copestake (1999), although this suggestion remains to be investigated in detail.

6.2 Augmentations to the constraint language

Our experience so far suggests that a rather rich constraint language is necessary in order to capture different types of underspecification. The qeq constraints that RMRS inherits from MRS are useful for underspecifying quantifier scope, but alternative machinery will be needed for PP-attachment ambiguities. Yet another type of constraint is required in principle for argument attachment with the RASP system, on the assumption that the semantic rules are mapped to the syntactic rules.

The problem can be illustrated by the following examples:

(10) to which dog did Kim give the bone?

[4]Inequalities in RMRS have strong similarities with inequalities in typed feature structure frameworks: see Carpenter (1992), for instance. In fact, what is under discussion here is another reflection of a standard theoretical problem for constraint-based approaches to grammar. In the current context, however, there's also a direct practical problem: i.e., how to avoid spurious matches between partially specified RMRSs.

[5]This issue may have to be revisited if we consider further specialising ERG semantics with anaphoric links, but there are several reasons to think that these will also best be treated by adding relations rather than equating variables directly.

(11) which bone did Kim give the dog?

The ERG would assign a semantics to these sentences where the indirect object is an ARG3 and the direct object an ARG2. It is clear, however, that when parsing *give the bone* and *give the dog* that it is not possible to distinguish between these possibilities. Obviously we could underspecify the argument as ARG2-3, but this would also have to apply to *chase the dog*, since we have no way of distinguishing between *chase* and *give* on the basis of limited lexical information. However, in most cases, the underspecification could be resolved once the complete sentence is known. For instance, in the example below, *dog* has to correspond to an ARG2 (ignoring ellipsis) whatever the verb X is.

(12) Kim Xed the dog

A formally correct solution to this would be to adopt a richer constraint language, with argument relations being variables, such as ARGN and ARGM, and with constraints being specifiable on the variables (e.g., ARGN $<$ ARGM). There would also be global constraints, preventing repeated argument relations, for instance.

However, for the time being we intend to ignore this issue and to overspecify in the cases like those described above. Examples where this is likely to be a problem appear relatively rare and since the RASP grammar does not fully cover the syntax of questions and similar constructions, they are likely to be misparsed in any case.

7 Using RMRS

This section contains a range of proposals for methods of integrating deep and shallow processing via RMRS.

7.1 Question answering

A quite straightforward use of RMRS which has been tried in a preliminary fashion is as part of a question-answering system.[6] The answer texts were shallow parsed with the RASP system, partial RMRSs were extracted, and compared with RMRSs extracted from the questions via the LinGO ERG. This involves a robust, weighted comparison between the RMRSs.

7.2 Incremental invocation of deep parsing on identified chunks of text

In some cases, we can identify ‘interesting’ subparts of text by means of specific cue phrase. This was mentioned in the introduction in the context of Teufel’s work, but the example used here concerns IE. For instance, suppose as part of an IE system, we are looking for comparative judgements about cell phones. The following sentence can be identified as potentially relevant, because it contains the cue *I prefer X to Y*:

(13) I really prefer the Nokia to the Motorola because it’s about 20 grams lighter.

While it is implausible that we can fully deep parse all the sentences in an IE application, because of parsing speed and the work needed to tune to each new domain, it is reasonable that we should be able to deep parse the cues, since these will be of restricted length and less domain dependent. For instance, *prefer* is likely to be used in a similar way whether talking about cell phones, motorcycles or wine. It is thus plausible that we can manually create a suitable lexicon for the cue texts. The reason for deep parsing is that we want to be able to identify arguments more precisely than can be done with shallow parsing alone (for instance, in the example above, we want to be able to utilise information about the subcategorization of *prefer*) and because we want to avoid false positives (for instance, where an adverb conveying negation is used).

The proposed methodology is to construct a library of cues expressed using RMRS, and to use this library to select regions of shallow-parsed text for further deep processing. For instance, in the example above, the cue phrase could be expressed as follows:

$$l0{:}\text{\_prefer\_v}(e), \text{ARG1}(l0, x1), \text{ARG2}(l0, x2), \text{ARG3}(l0, x3), l1{:}\text{pron}_{1st}(x1) \rightarrow \text{BETTER}(x2), \text{WORSE}(x3)$$

The left hand side of the rule will match some piece of text, while the right hand side is intended to be indicative of some action that might result from recognition of the cue, such as populating some database. These sort of rules can be regarded either as a form of pattern-matching on semantic representations, or as a variety of shallow inference: the ‘munging’ rule machinery in the LKB currently supports something similar on MRSs.

[6]These experiments have been done as part of the language practicals in the Cambridge Masters course in Computer Speech Text and Internet Technology. The practicals are run by Ted Briscoe, Simone Teufel and the author.

A shallow parser that output an RMRS structure would serve to identify a possible cue. For instance, assume that a POS tagger plus NP chunker was being used on the example above. The left hand side of the cue would unify with the RMRS extracted by the shallow parser, so matching it. The deep parser can then be invoked on the textual region identified by the cue. The interface between the shallow and deep parser is the original text aligned with the indexed RMRS, so the cue can be associated with a specific region of text (possibly non-contiguous, if parenthetical information is excluded). The deep parser would confirm that the cue matched and would identify the two relevant arguments to *prefer*, enabling the database to be populated.

The most plausible way of building the RMRS cue library is to hand-annotate some sentences to identify cue phrases, deep parse the cues to produce RMRSs (with manual checking/parse selection) and then to have a semi-automatic method of generalising over sets of cues.

7.3 Patching up failed deep parses

The compatibility of composition between deep and shallow processing also makes RMRS a promising approach for connecting fragmentary deep parses. The idea is to take the phrases that can be deep analysed, and connect them via the semantics given by the shallower analyser, using structures with compatible bracketing. For instance, suppose the deep analyser has produced an analysis of the following sentence with the bracketed components analysed, but without a complete parse being found, perhaps because the grammar does not allow adverbs before subcategorized arguments:

(14) (The American consultant)$_{np}$ (expected)$_{v}$ (us)$_{np}$ (unknowingly and inadvertently)$_{adv}$ (to compromise the integrity of our software)$_{vp}$.

A shallow parser that had fewer constraints might analyse this sentence, attaching the *to* phrase to the verb with an ARGN relationship. This analysis would be compatible with the information from the structures produced by the deep parser, allowing those to be connected. In fact, given reasonable semantic constraints on the relation for *expect*, the ARGN could be specialised to ARG3.

It is currently unclear whether the semantics output by, say, the RASP parser could be made sufficiently tightly compatible with the ERG to allow this approach to work. This approach would probably require the semantics for closed-class words to be largely consistent between RASP and the ERG, which requires that part of the semantic interface to the ERG (the SEM-I: Copestake and Flickinger, 2003) be in place.

7.4 Unknown words/phrases

RMRS might also be useful as part of an alternative to an unknown word/phrase mechanism for deep parsing. If a shallow parser can identify a phrase as a particular sort of named entity, for example, the deep parser can incorporate a structure corresponding to that phrase in the parse. For instance, consider the following example:

(15) Through howling wind and freezing rain the tourists keep coming to the place where Johnny 'Mad Dog' Adair used to live.

Assume that the deep parser fails on *Johnny 'Mad Dog' Adair* but that a named-entity recogniser can analyse it. A basic template for an NP allows a phrasal sign to be constructed which can be syntactically incorporated into the parse: if the named entity recogniser can produce RMRS structures, then the semantics will also be coherent. The main question for the approach is whether the RMRS that is produced by the named entity recogniser should be equivalent to something that would be constructed by the deep parser if the phrase could be analysed (tight integration) or simply a structure that gives rise to a well-formed semantic result, potentially with different internal information than would arise from the deep parsing process, if that were to succeed. For example, as part of an unknown word mechanism, a named entity system might produce an RMRS such as:

$$[h0, x][h0\text{:def}(x, h1, h2), h4\text{:person-name}(x), \text{NAME}(h4, `JohnnyAdair'), \text{NICKNAME}(h4, `MadDog')]$$

The use of the quantifier means that this is structurally compatible with the MRS that would be produced by the ERG, but the ERG could not easily be modified to construct a comparable internal structure. It remains to be seen whether tight integration is worth pursuing in this case.

Named entity recognition would also be a useful refinement to the idea of cue phrase matching, described above. Some regions of the text passed to the deep-parser could be marked-up by a named-entity recogniser associated with shallow parsing. The deep parser then does not need to access the internal structure of these phrases, but can instead simply incorporate them into the result.

7.5 Parser evaluation

Briscoe et al (2002) suggest that RMRS might be useful as an output format for parser evaluation. Since RMRSs are maximally factorised, calculating the degree of overlap between a parser output and some gold standard is possible. For this to be viable, there needs to be agreement about the gold standard account of a wide range of phenomena, so eventually this would require something along the lines of the SEM-I described in Copestake and Flickinger (2003).

8 Conclusions and further work

The main idea proposed here is that we can manipulate the syntax of the semantic representation so it can be split into minimal composition units suitable for a variety of processors. By factoring the semantics into minimal units, we have a basis for comparison between systems that can produce differing amounts of semantic information. The formalism allows for radical underspecification of semantic content. So far this looks promising, since many ideas seem to flow from this.

RMRS is under active development. For current work, please see the DELPH-IN web site: www.delph-in.net

A RMRS DTD

Version of 26/01/2005: current as of 23/06/2006

```
<!ELEMENT rmrs-list (rmrs)*>
<!-- Documentation in dtd-notes.txt -->
<!ELEMENT rmrs (label, (ep|rarg|ing|hcons)*)>
<!ATTLIST rmrs
          cfrom CDATA #REQUIRED
          cto   CDATA #REQUIRED
          surface   CDATA #IMPLIED
          ident     CDATA #IMPLIED >

<!ELEMENT ep ((realpred|gpred), label, var)>
<!ATTLIST ep
          cfrom CDATA #REQUIRED
          cto   CDATA #REQUIRED
          surface   CDATA #IMPLIED
          base      CDATA #IMPLIED >

<!ELEMENT realpred EMPTY>

<!ATTLIST realpred
          lemma CDATA #REQUIRED
          pos (v|n|j|r|p|q|c|x|u|a|s) #REQUIRED
          sense CDATA #IMPLIED >

<!ELEMENT gpred (#PCDATA)>

<!ELEMENT label EMPTY>

<!ATTLIST label
          vid CDATA #REQUIRED >

<!ELEMENT var EMPTY>
<!ATTLIST var
          sort (x|e|h|u|l) #REQUIRED
          vid  CDATA #REQUIRED
          num  (sg|pl|u) #IMPLIED
          pers (1|2|3|1-or-3|u) #IMPLIED
          gender (m|f|n|m-or-f|u) #IMPLIED
          divisible (plus|minus|u) #IMPLIED
          cogn-st (type-id|uniq-id|fam|activ|in-foc|uniq-or-less|uniq-or-fam|
fam-or-activ|active-or-more|fam-or-less|uniq-or-fam-or-activ|fam-or-more|
activ-or-less|uniq-or-more|u) #IMPLIED
          tense (past|present|future|non-past|u) #IMPLIED
          telic (plus|minus|u) #IMPLIED
          protracted (plus|minus|u) #IMPLIED
          stative (plus|minus|u) #IMPLIED
          incept (plus|minus|u) #IMPLIED
          imr (plus|minus|u) #IMPLIED
          boundedness (plus|minus|u) #IMPLIED
          refdistinct (plus|minus|u) #IMPLIED >


<!ELEMENT rarg (rargname, label, (var|constant))>

<!ELEMENT rargname (#PCDATA)>
```

```
<!ELEMENT constant (#PCDATA)>

<!ELEMENT ing (ing-a, ing-b)>
<!ELEMENT ing-a (var)>
<!ELEMENT ing-b (var)>

<!ELEMENT hcons (hi, lo)>
<!ATTLIST hcons
          hreln (qeq|lheq|outscopes) #REQUIRED >

<!ELEMENT hi (var)>
<!ELEMENT lo (label|var)>
```

Notes on the DTD:

- POS is lowercase and obligatory:

  ```
  (v|n|j|r|p|q|c|x|u|a|s)
  v - verb
  n - noun
  a - adjective or adverb (i.e. supertype of j and r)
  j - adjective
  r - adverb
  s - verbal noun (used in Japanese and Korean)
  p - preposition
  q - quantifier (needs to be distinguished for scoping code)
  x - other closed class
  u - unknown

  The implicit hierarchy is:
   n := u
   v := u
   a := u
   j := a
   r := a
   s := n, s:= v
   p := u
   q := u
   c := u
   x := u
  ```

- Labels are always implicitly of the same sort (i.e., label)
- cfrom and cto default to -1 if unknown
- The surface attribute on rmrs and ep is used to give the original surface string. It is optional.
- The base attribute is used for languages where the current parsing engines cannot represent the lemma properly. For example, in Jacy the lemma is in ASCII, but the base form should often be Chinese characters. This attribute is depreciated - as the lemma attribute in realpred is a string, in theory it can handle the real base form (we can use all of unicode). When all the tools can handle it, then we can remove it. For the moment, base should only be added if it is different from both lemma and string.
- ident is an attribute on rmrs's to identify which utterance they belong with. The HoG currently uses a wrapper around the RMRS, with identifying information there instead. Hinoki uses the ident identifier but may switch to a wrapper, in which case ident may be removed. In any case it is optional.

B References

Bos, Johan (1995) ‘Predicate Logic Unplugged’, *Proceedings of the Tenth Amsterdam Colloquium,* Amsterdam, Holland.

Briscoe, Ted, John Carroll, Jonathan Graham, Ann Copestake (2002) ‘Relational Evaluation Schemes’, *Proceedings of the Beyond PARSEVAL Workshop at LREC 2002,* Las Palmas, Gran Canaria, pp. 4–8.

Briscoe, Ted and John Carroll (2002) ‘Robust accurate statistical annotation of general text’, *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002),* Las Palmas, Gran Canaria.

Carpenter, R. (1992) *The Logic of Typed Feature Structures,* Cambridge Tracts in Theoretical Computer Science No. 32. Cambridge University Press, New York.

Carroll, John, Ted Briscoe, and Antonio Sanfilippo (1998) ‘Parser evaluation: a survey and a new proposal’, *Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC-1998),* Granada, Spain, pp. 447–454.

Copestake, Ann, Dan Flickinger, Ivan A. Sag and Carl Pollard (2005) ‘Minimal Recursion Semantics: An introduction’, *Research on Language and Computation,* **3(2–3)**, 281–332.

Copestake, Ann and Dan Flickinger (2000) ‘An open-source grammar development environment and broad-coverage English grammar using HPSG’, *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000),* Athens, Greece, pp. 591–600.

Copestake, Ann, Alex Lascarides and Dan Flickinger (2001) ‘An Algebra for Semantic Construction in Constraint-based Grammars’, *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001),* Toulouse, France.

Copestake, Ann and Dan Flickinger (2003) ‘The semantic interface to the ERG and other LinGO grammars’, Deep Thought website.

Crysmann, Berthold, Anette Frank, Bernd Kiefer, Stefan Mueller, Guenter Neumann, Jakub Piskorski, Ulrich Schaefer, Melanie Siegel, Hans Uszkoreit, Feiyu Xu, Markus Becker and Hans-Ulrich Krieger (2002) ‘An integrated architecture for shallow and deep processing’, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002),* University of Pennsylvania, Philadelphia, USA, pp. 441–449.

Daum, Michael, Kilian Foth and Wolfgang Menzel (2003) ‘Constraint Based Integration of Deep and Shallow Processing Techniques’, *Proceedings of the 10th Meeting of the European Chapter of the Association for Computational Linguistics (EACL-2003),* Budapest, Hungary.

Dowty, David (1989) ‘On the semantic content of the notion ‘thematic role” in Gennaro Chierchia, Barbara Partee and Ray Turner (ed.), *Property Theory, Type Theory and Natural Language Semantics,* Reidel, Dordrecht, The Netherlands, pp. 69–129.

Kasper, W., Kiefer, B., Krieger, H., Rupp, C., Worm, K. (1999) ‘Charting the depths of robust speech parsing’, *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL ’99),* College Park, Maryland, USA.

Lascarides, Alex and Ann Copestake (1999) ‘Default representation in constraint-based frameworks’, *Computational Linguistics,* **25.1**, 55–106.

Stephan Oepen, Daniel Flickinger, Hans Uszkoreit and J-I. Tsujii (2003) *Collaborative Language Engineering: A Case Study in Efficient Grammar-based Processing,* CSLI Publications.

Parsons, T. (1990) *Events in the Semantics of English,* MIT Press.

Anna Ritchie (2004) ‘Compatible RMRS Representations from RASP and the ERG’, University of Cambridge Computer Laboratory, technical report TR-661, Project deliverable for DeepThought, `http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-661.html`.

Teufel, Simone (1999) *Argumentative Zoning: Information Extraction from Scientific Text,* PhD thesis: School of Cognitive Science, University of Edinburgh, Edinburgh, UK.