Resources for Building Applications with Dependency Minimal Recursion Semantics

Ann Copestake*, Guy Emerson*, Michael Wayne Goodman†, Matic Horvat*, Alexander Kuhnle*, Ewa Muszyńska*

* University of Cambridge, Cambridge, UK
{aac10, gete2, mh693, aok25, emm68}@cam.ac.uk
† University of Washington Seattle, WA, USA
goodmami@uw.edu

Abstract

We describe resources aimed at increasing the usability of the semantic representations utilized within the DELPH-IN (Deep Linguistic Processing with HPSG) consortium. We concentrate in particular on the Dependency Minimal Recursion Semantics (DMRS) formalism, a graph-based representation designed for compositional semantic representation with deep grammars. Our main focus is on English, and specifically English Resource Semantics (ERS) as used in the English Resource Grammar. We first give an introduction to ERS and DMRS and a brief overview of some existing resources and then describe in detail a new repository which has been developed to simplify the use of ERS/DMRS. We explain a number of operations on DMRS graphs which our repository supports, with sketches of the algorithms, and illustrate how these operations can be exploited in application building. We believe that this work will aid researchers to exploit the rich and effective but complex DELPH-IN resources.

Keywords: English Resource Semantics, Operations on Semantics, Tools for Semantics

1. Introduction

DELPH-IN (`http://www.delph-in.net`) is an international collaboration between researchers with an interest in 'deep', linguistically-motivated, language processing. Several broad-coverage computational grammars have been produced, together with experimental grammars for languages in all the main language families. These grammars share a framework for compositional meaning representation known as Minimal Recursion Semantics (MRS; Copestake et al. (2005)), which has several variants, collectively referred to as *MRS. Currently, the most used and widest coverage DELPH-IN grammar is the English Resource Grammar (ERG; Flickinger (2000)).

*MRS from the ERG (henceforth English Resource Semantics: ERS) and other DELPH-IN grammars has been successfully used in a number of applications. These include machine translation (e.g., Bond et al. (2011)), information extraction and question answering (e.g., Copestake et al. (2006), Frank et al. (2005), MacKinlay et al. (2009), MacKinlay et al. (2012)), extraction of ontological relationships (e.g., Herbelot and Copestake (2006)), question generation (e.g., Yao and Zhang (2010), Yao et al. (2012)), entailment recognition (e.g., Lien and Kouylekov (2014)), detection of scope of negation (e.g., Packard et al. (2014)) and natural language interfaces (e.g., Packard (2014)). In several cases (including Packard et al. (2014), Lien and Kouylekov (2014), Packard (2014), Yao and Zhang (2010), MacKinlay et al. (2009)) an ERS-based system has outperformed other approaches in a competitive task. While an ERG-based system is not suitable for applications which require rapid analysis of very large quantities of text, parsing accuracy is competitive with other approaches and is generally good on new domains without any need for substantial training or adaptation (MacKinlay et al. (2010), Dridan and Oepen (2013)). ERS is especially suitable in applications which require high precision in semantic tasks and where some lack of recall due to parse failure rates is tolerable. Furthermore, the ERG is bidirectional and ERSs can also be used for realization.

However, the adoption of ERS (and *MRS with other grammars) has been limited outside the DELPH-IN community, despite the fact that the resources are all Open Source. Anecdotally, a major problem is the complexity of the analyses. The DELPH-IN community have taken various steps towards addressing this. There is a standardized semantic interface to the grammars (the SEM-I: Flickinger et al. (2005)). Flickinger et al. (2014) described an approach to documenting the fundamentals of ERS, allowing users to look up the detailed analysis of particular grammatical constructions. The Elementary Dependency representation (Oepen and Lønning, 2006) is simpler than *MRS and captures many aspects of ERS while being relatively similar to familiar syntactic dependency formalisms. It can be used for comparative parser evaluation as well as other applications (Ivanova et al. (2012), Dridan and Oepen (2011)).

In this paper, we concentrate on resources for processing Dependency MRS (DMRS; Copestake (2009)). This is a version of MRS which can be represented as a dependency graph, with no need for variables. Unlike Elementary Dependencies, DMRS has a proper representation of scope and is fully inter-convertible with other forms of *MRS. It thus makes use of the full power of the ERG for semantic description. DMRS has been used in a number of applications (e.g., Reiplinger et al. (2012), Schäfer et al. (2011), Herbelot (2013), Horvat et al. (2015), Emerson and Copestake (2015)) and our aim is to increase its use.

There have, of course, been earlier graph-based semantic representation languages, including some which support genuine logical forms (e.g., Allen et al. (2008)). The popularity of Abstract Meaning Representation (AMR; Banarescu et al. (2013)) has led to increased interest in graph-based formalisms. Bender et al. (2015) discusses why compositionality (as assumed in *MRS but not in AMR) is an

important property of semantic representations. We believe that the DMRS is interestingly different from these other representations and that the coverage and depth of ERS have significant potential advantages in applications, if researchers can overcome the difficulties arising from its detail and complexity.

Our purpose here is thus to improve the utility of DMRS to encourage increased uptake of DELPH-IN resources. In this paper, we start by outlining DMRS and some existing resources and then move on to describe a repository which supports developers by providing standard operations on DMRS and giving illustrative examples of their potential use in applications.

2. An Outline of DMRS

Rather than give a formal introduction to DMRS, we will outline its properties by means of the example shown in Fig 1. This shows a scoped structure in predicate calculus using generalized quantifiers, the corresponding MRS, the Robust MRS (RMRS; Copestake (2007b)), and finally the DMRS. Both MRS and RMRS can be regarded as a flat list of elementary predications, unlike a conventional logical representation which uses nesting to convey scope. The labels (l1, l2 etc) shown in the MRS and RMRS structures together with the qeq constraints represent (underspecified) scope restrictions (e.g., the restrictor of *every*). The ERG actually uses argument labels internally (RSTR, ARG1 etc), but to make the comparison with the scoped representation clearer, the MRS in Figure 1 is shown in a format which omits them. RMRS always makes the argument labels explicit: arguments may be dropped or underspecified without the RMRS structure being ill-formed, but the cost is that an additional anchor element (a1, a2 etc) is required to relate arguments to the predications.

The DMRS is essentially a simplification of the complex graph which results if identities between labels and variables in the RMRS are replaced by links, with the canonical DMRS form being produced by exploiting general conditions of ERS. These mean, for instance, that the RMRS ARG0 is not needed.

One complication discussed by Copestake (2009) is that occasionally undirected bare /EQ links are necessary to completely model the MRS. This situation arises when two elementary predications have equal labels in the MRS/RMRS but there is no argument relation between them: e.g., in some (fairly unusual) types of relative clause. We are currently seeking to revise the definition of DMRS to avoid such undirected links, but since they do not substantially affect the main objectives of this paper, we will not discuss that further here.

Although a more detailed description of DMRS and its semantics is out of the scope of this paper, it will be useful in what follows to note that there are several different notions of DMRS well-formedness:

- Realizability: A DMRS meets the realizability constraint with respect to a grammar if it corresponds to a natural language string according to that grammar (for the current ERG, this is mediated by the MRS). Of course, not all semantically-meaningful DMRSs meet this condition.
- MRS well-formedness: A DMRS meets this condition if it can be converted into a well-formed MRS as defined by Copestake et al. (2005). Informally, a well-formed MRS is one that can be specialized into one or more scope-resolved structures. For a grammar where all semantic composition operations and lexical entries are correctly specified, all MRS structures produced by/accepted by the grammar for an utterance will meet this well-formedness condition,[1] and hence the set of DMRSs which are well-formed under this condition are a subset of those which meet the realizability constraint. This condition is also a requirement for conversion into predicate calculus.
- Basic well-formedness: simply requires that the DMRS is a graph with well-formed labels on nodes and edges. If the bare EQ links described above are excluded, the graph must be acyclic, but not necessarily connected.

3. DMRS Resources

Links to resources for DMRS, including those highlighted below, can be found on `http://moin.delph-in.net/RmrsDmrs`. DELPH-IN is committed to the development of Open Source resources, and all software listed below (including the ERG, ACE, pyDelphin, and pydmrs) uses the MIT license. The HPSG annotations in the WikiWoods corpus are covered by the LGPL v3 license.

3.1. The English Resource Grammar and DELPH-IN Tools

*MRSs are produced from English text with the ERG using suitable preprocessing tools and a compatible parser. Details are beyond the scope of this paper, but see `http://moin.delph-in.net/`. DELPH-IN realizers allow text to be produced from *MRS input. For most of our work, we now use Woodley Packard's Answer Constraint Engine (ACE: `http://sweaglesw.org/linguistics/ace/`) for parsing and realization.

3.2. pyDelphin

The pyDelphin library (`https://github.com/delph-in/pydelphin`) is an Open Source Python package implementing a variety of DELPH-IN formal representations, with a particular emphasis on MRS and DMRS support. As most of the parsers/realizers which handle DELPH-IN grammars (including ACE) currently assume MRS output/input rather than DMRS, pyDelphin's MRS⇌DMRS conversion functionality is useful for researchers wishing to work with DMRS representations. A Python wrapper for ACE is also provided, which can simplify applications or workflows that rely on ACE.

3.3. WikiWoods DMRS

WikiWoods (Flickinger et al., 2010) is a dump of Wikipedia parsed with the ERG. It is very useful for applications where a large number of semantic contexts are required, for

[1] Note that in the ERG treatment of fragments, an additional predication is associated with an utterance so it can be resolved into a scope-resolved structure (see Fig 2).

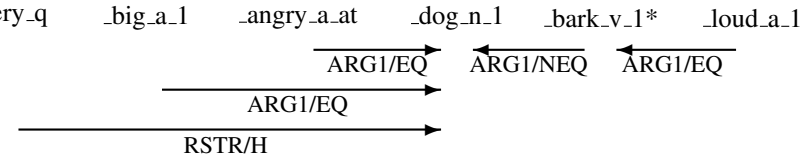
Sentence:	*Every big angry dog barks loudly.*
Scoped form (single scope for this example):	_every_q(x4, _big_a_1(e8,x4) ∧ _angry_a_1(e9, x4) ∧ _dog_n_1(x4), _bark_v_1(e2,x4) ∧ _loud_a_1(e10,e2))
MRS (hiding argument labels):	l1:_every_q(x4,h5,h6), h5 qeq l2, l2:_big_a_1(e8,x4), l2:_angry_a_1(e9,x4), l2:_dog_n_1(x4), l4:_bark_v_1(e2,x4), l4:_loud_a_1(e10,e2)
RMRS:	l1:a1:_every_q, BV(a1,x4), RSTR(a1,h5), BODY(a1,h6), h5 qeq l2, l2:a2:_big_a_1(e8), ARG1(a2,x4), l2:a3:_angry_a_1(e9), ARG1(a3,x4), l2:a4:_dog_n_1(x4), l4:a5:_bark_v_1(e2), ARG1(a5,x4), l4:a6:_loud_a_1(e10), ARG1(a6,e2)
DMRS:	

Figure 1: Semantic representations for 'Every big angry dog barks loudly.' Note that the directionality of the modifier links in the DMRS follows from the predicate calculus representation but is the opposite of conventional syntactic dependencies. Tense and number information is associated with variables in MRS and RMRS and nodes in DMRS, but is omitted here for clarity. Figure 2 shows a complete representation.

```
<dmrs surface="Allusions to other works" ident="1214528100230">
   <node nodeid="10001" cfrom="2" cto="26">
      <gpred>unknown_rel</gpred><sortinfo cvarsort="e" sf="prop-or-ques"/>
   </node>
   <node nodeid="10002" cfrom="2" cto="26">
      <gpred>udef_q_rel</gpred><sortinfo/>
   </node>
   <node nodeid="10003" cfrom="2" cto="11">
      <realpred lemma="allusion" pos="n" sense="1"/>
      <sortinfo cvarsort="x" pers="3" num="pl"/>
   </node>
   <node nodeid="10004" cfrom="12" cto="14">
      <realpred lemma="to" pos="p"/>
      <sortinfo cvarsort="e" sf="prop" tense="untensed" mood="indicative"/>
   </node>
   <node nodeid="10005" cfrom="15" cto="26">
      <gpred>udef_q_rel</gpred><sortinfo/>
   </node>
   <node nodeid="10006" cfrom="15" cto="20">
      <realpred lemma="other" pos="a" sense="1"/>
      <sortinfo cvarsort="e" sf="prop" tense="untensed" mood="indicative"/>
   </node>
   <node nodeid="10007" cfrom="21" cto="26">
      <realpred lemma="work" pos="n" sense="1"/>
      <sortinfo cvarsort="x" pers="3" num="pl"/>
   </node>
   <link from="10001" to="10003"><rargname>ARG</rargname><post>NEQ</post></link>
   <link from="10002" to="10003"><rargname>RSTR</rargname><post>H</post></link>
   <link from="10004" to="10003"><rargname>ARG1</rargname><post>EQ</post></link>
   <link from="10004" to="10007"><rargname>ARG2</rargname><post>NEQ</post></link>
   <link from="10005" to="10007"><rargname>RSTR</rargname><post>H</post></link>
   <link from="10006" to="10007"><rargname>ARG1</rargname><post>EQ</post></link>
</dmrs>
```

Figure 2: An example of DMRS in XML format, from the WikiWoods corpus. This format is intended to be machine-readable rather than human-readable. It includes information about number and tense (omitted for simplicity in Figure 1) and also shows the surface string and character position information essential for bookkeeping. Note the 'UNKNOWN_REL' which means this fragment can be treated as a proposition.

instance building distributional models from parsed data. The DMRS version of WikiWoods has been used by a number of researchers (e.g., Herbelot (2013), Emerson and Copestake (2015)) since it is much easier to extract semantic relationships from it than to deal with variable interactions in MRS/RMRS. An example WikiWoods DMRS is shown in Fig 2.

3.4. DMRS Application Libraries: pydmrs

The bulk of this paper concerns pydmrs: a new Python package supporting operations such as DMRS subgraph matching, standard types of DMRS simplification, mapping and chunking as discussed in more detail in §4. A user may choose either to import the routines provided in this package into their own Python code, or to run specific modules as batch scripts to process large amounts of data. Detailed examples of various uses are given in the pydmrs repository. We are also including some sample applications, outlined in §5, in order to help developers to investigate these resources further themselves.

4. Operations on DMRS

By considering the DMRS as a semantic graph, we define a number of operations which can be utilized in applications. Our aim here is to motivate the abstract operation types and outline how they can be exploited. Note that all these types of operation are applicable both where machine learning is used and for hand-written rules or patterns.

4.1. Matching

Matching is the basic *MRS operation used in information extraction and question answering. Copestake (2007a) discusses how the *MRS matching operation can be regarded as a form of robust inference. In the robust approach to realization described by Horvat et al. (2015), subgraph matching is required to find which rules can be applied to the input DMRS graph. Matching can be used with wildcards to implement ontology extraction, as outlined in §5. One advantage of matching using DMRS, as opposed to the earlier *MRS approaches, is that DMRS structures may be packed, allowing efficient matching operations to be performed on a forest of alternative structures.

The basic operation of DMRS matching amounts to subgraph matching, with conditions on node and arc labels that they are compatible rather than identical. It can thus be stated more simply than MRS/RMRS matching and is more straightforward to implement, since there is no need to maintain a list of variable equalities. *MRS matching is worst-case exponential, but, as discussed below, efficient algorithms exist for practical uses.

The pydmrs library includes three algorithms for DMRS matching: exact subgraph matching, surface-aligned matching and general scored matching.

The **exact (sub)graph matching** checks whether a DMRS graph contains a specified subgraph. If any element of the subgraph is missing in the larger graph, no match is detected. The efficiency of this matching method is based on the assumption that a DMRS graph corresponding to a sentence or a sentence fragment mainly consists of uniquely-labelled nodes, which hence match unambiguously. A naive matching algorithm performing an exhaustive search over all possible node match combinations, with worst-case exponential runtime, is therefore sufficient in practice. The algorithm identifies the unambiguous node matches and then adds all matches that can be inferred for repeated labels by considering their immediately neighbouring nodes. Node labels which are repeated in a sentence will generally correspond to common words like quantifiers or prepositions, which will be connected to open class words which are unlikely to be repeated. If all of the neighbouring nodes already match uniquely and there is only one node in the other graph containing the corresponding neighbourhood, we can conclude a match of the node in question. Subsequently, we continue to perform an exhaustive search over all the possible combinations if there are still unmatched nodes left.

The other two matching approaches find the best possible match, even a non-exact one, and assess its quality with an F-score, computed based on the number of matched nodes and links, similarly to a metric by Cai and Knight (2013). Nodes and links are treated as equally important.

The **surface-aligned matching** uses the fact that parser output contains information on which part of the original string corresponds to a given predicate. By ordering the DMRS nodes by the position of their predicates in a sentence, we can limit the search space of the matching algorithm. There are two ways to use this form of matching. The matched subgraph returned as the result of the query can include only nodes from the query, or it can include all the nodes corresponding to the aligned region of the surface. For example, if we query for *big dog* in the DMRS graph from Figure 1, the first variant of the algorithm grants the match a perfect score. In the second variant the match is penalized because the surface region of interest and the matched subgraph include *angry*.

The above approach cannot be used when the alignment information is missing. Moreover, if two different sentences have the same DMRS representation but different word order, the algorithm will not match them perfectly. To account for these and other scenarios, we provide an alternative in the form of a less efficient **general matching algorithm**. The algorithm traverses the DMRS graph following its link structure exploring any matching regions and combining the disconnected matches from individual regions in a way which maximizes the score. The resulting subgraph contains only the nodes and links from the query.

4.2. Simplification

For many applications, much simpler structures than the full ERS are all that is required. For instance, for an application involving a simple interface to a robot, quantifiers may not be needed. However, there is little point in providing a single 'Simplified MRS' as an alternative to the standard ERS. Quantifiers are not needed for some applications, but are essential for others. Our approach, therefore, is to define a series of standard simplification operations which can be combined as required for any given application.

We define simplification as any DMRS graph rewriting operation which results in a reduction of complexity by removal of particular classes of subgraph. In some cases, a

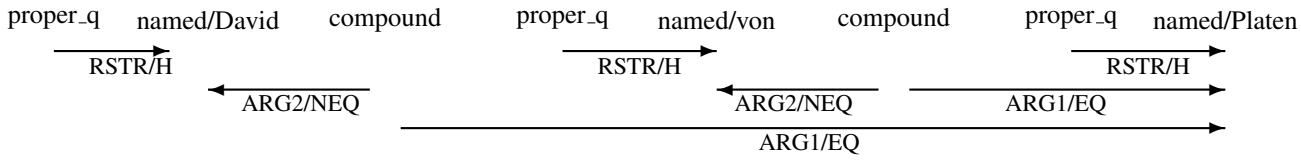


Figure 3: Original DMRS/ERS structure for the name *David von Platen*.

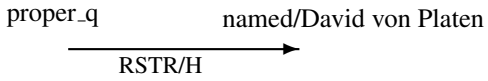


Figure 4: Simplified DMRS/ERS structure for *David von Platen*

subgraph may be replaced with a smaller subgraph. We distinguish between reversible, semi-reversible and lossy simplifications: after lossy simplification the DMRS is not required to meet any well-formedness criteria other than that of basic well-formedness.

4.2.1. Simplification of Complex ERS Structures

There are cases where the generality and compositionality of ERS produces structures which are inconveniently complex for many applications. For instance, numbers are represented compositionally: the DMRS for *twenty five* has three nodes — two nodes for numbers '20' and '5' and a 'plus' node connecting the two. Although this representation follows from MRS compositionality principles and is required for full generality (e.g., *three or four thousand* with the interpretation 3000 or 4000), the representation of numbers written using words can often usefully be simplified to be equivalent to the representation using digits. For instance, the simplified DMRS for *twenty five* is equivalent to the DMRS for *25*.

We provide a number of operations to simplify complex structures of this type. Consider the case of proper names such as *David von Platen*. The ERS for this structure is given in Fig. 3, with the simplified structure in Fig. 4.

As shown, in the simplified structure, the name is treated as a single element, as though it were a simple proper name (see below for discussion of PROPER_Q). As with numbers, there are cases where this simplification process fails (e.g., *David and Albrecht von Platen*), which justifies the ERS approach as a more general account, but for most texts, such examples are rare.

Complex structure simplification may optionally be followed by grammar predicate filtering (discussed below), in which case the PROPER_Q can also be removed.

In general, complex structure simplification is semi-reversible. A *MRS can be constructed from the simplified DMRS which is equivalent to an ERS structure, but the precise input structure may not be recoverable. For instance, names with more than two elements have multiple bracketing possibilities in ERS: name simplification can be reversed to a canonical structure equivalent to one of these, but cannot recover the original analysis. Complex structure simplifications nevertheless meet the realizability and MRS well-formedness conditions (§2).

4.2.2. Grammar Predicate Filtering

In addition to lexically-derived predicates, ERS uses a considerable number of grammar predicates, such as PROPER_Q and COMPOUND. Grammar predicates capture the semantic contribution of various grammatical constructions. For instance, COMPOUND is used to represent compound nouns (e.g., *kitchen knife*) as well as the complex names discussed above.

Although grammar predicate nodes are an important facet of ERS, they are not needed for some applications. For example, although PROPER_Q is necessary in order to produce a well-formed structure according to the MRS criteria, it can usually be omitted without losing information. Another example is ELLIPSIS, which allows a representation of the semantics of elliptical constructions, but in practice very often occurs in misparses. Such nodes increase the size and complexity of the DMRS graph, and also make it more difficult to align the DMRS with the original text: developers working with DMRS may therefore remove them. For example, in graph to string realization (Horvat et al., 2015), unneeded grammar predicate nodes are removed from the DMRS graph to lessen the data sparsity problem with translation grammar, as well as to speed up the otherwise resource and time intensive decoding.

Our current aim is to make removal of predicate nodes less ad hoc. Grammar predicate filtering works by matching individual nodes against a filter list and, when a match occurs, removing the node and adjacent edges from the DMRS graph. Specifying which grammar predicates to filter is accomplished using a configuration file. We provide a default configuration used by Horvat et al. (2015) in pydmrs.

Removing some types of grammar predicate nodes may result in a disconnected graph. Since this is generally undesirable, we implemented optional functionality that only removes a grammar predicate node if the graph will remain connected. Computing whether a graph is disconnected can be done with breadth-first search: starting from any node, we iteratively visit adjacent nodes; if any nodes are left unvisited at the end, the graph is disconnected.

4.3. Conversion to Rooted Acyclic Graphs

There are many useful algorithms operating over trees, which pass either from the root to the leaves, or vice versa. For example, Socher et al. (2010) present a recursive neural network architecture, where the leaves of a syntactic parse tree are associated with embedding vectors, which are recursively combined to give vector representations of phrases, until we reach the root of the tree. Such algorithms are difficult to extend to an arbitrary directed graph, as there may be cycles or multiple roots. Although DMRS inherently requires non-tree structures for full expressivity (e.g.

Search: `[2]:node? <-3- [1]:_give_v_1 e? -2-> _hand_n_1 x? <-- _a_q`
Replace: `[1]:_help_v_1 e? -2-> [2]:node?`

Example sentence: *"I can give you a hand with your work."* → *"I can help you with your work."*

Search: `[2]:node? <-1- [1]:_give_v_1 e? -3-> [3]:node?`
Replace: `[3]:node? <-1- [1]:_get_v_1 e? <=1= _from_p e -2-> [2]:node?`

Example sentence: *"Kim gave a book to Sandy."* → *"Sandy got a book from Kim."*

Search: `[1]:_like_v_1 e? -2h-> [2]:_?_v e[pui--]`
Replace: `[1]:_enjoy_v_1 e? -2h-> [2]:_?_v e[pui-+]`

Example sentence: *"I like to play tennis."* → *"I enjoy playing tennis."*

Figure 5: Example for paraphrases of the idiom *"give a hand"* to *"help"*, *"give"* to *"get"* (involving argument re-ordering) and *"like to"* to *"enjoy"* (involving a gerund; note the -/+ in the square bracket event variable specification indicating progressive form) using DMRS mapping (question marks denote underspecification). The subgraph patterns are written in our DMRS graph description language.

Kim promised to go), for many tree-based algorithms, it suffices to have a graph that is both acyclic and rooted, because it is then possible to pass from the root to the leaves, or vice versa. To facilitate using such algorithms, we provide functionality for converting DMRS structures to rooted acyclic graphs, by changing the directionality of certain links, such as modifiers. As a side effect, this results in a structure that is closer to syntactic dependencies.

4.4. Mapping

DMRS mapping is a two-step process, referring to the identification of a DMRS subgraph, followed by its replacement by another (i.e., a form of graph rewriting) to obtain a modified DMRS graph that can be used in realization.

First, the subgraph S to be replaced has to be identified in the DMRS graph G. This is done via a form of exact subgraph matching, where nodes in S are associated with corresponding nodes in G and *all* links in this subgraph of G match the ones in S (see §4.1). However, matching nodes exactly would be very inflexible and would require writing all possible concrete instantiations of a mapping pattern. We therefore introduce underspecification for the attributes of a node, like predicates and variable features. For example, this allows us to specify that a node must be nominal or verbal but to underspecify the tense of a verb.

The subgraph S found in G then is replaced by another subgraph S'. To do this, the mapping rule has to specify how the relevant *anchor* nodes of S in G are associated with respective nodes in S'. The anchor nodes of S' can again be underspecified, meaning that they overwrite only specific attributes of the corresponding node of S and leave the rest unchanged. For example, this allows us to keep the tense of a verb but change the lemma.

DMRS mapping has a variety of applications. Sets of subgraphs that are considered to be semantically equivalent (in a specific context) correspond to possible paraphrases, and mapping between them corresponds to the process of paraphrasing (see §5.2). By distinguishing one of the paraphrases as the target to which others are mapped, one can *"normalise"* sentences to a subset of possible expressions. Initial experiments with DMRS mapping for paraphrase using automatically acquired rules are described by MacKinlay (2012, ch7). On the other hand, when the source and target languages of the subgraphs are different, mapping becomes a method of machine translation, analogous to the use of MRS transfer rules in MT (compare e.g., Bond et al. (2011) or Oepen et al. (2004)). We are experimenting with ways of generalizing the current mapping procedure presented above. One application of interest is question generation (see Yao and Zhang (2010) and Yao et al. (2012)), which reduces to a more structural variant of DMRS mapping than straightforward node-to-node replacement.

4.5. Chunking

Chunking refers to the splitting of a DMRS graph into subgraphs which can be processed independently. The results of processing can be recombined with minimal loss of information. For instance, if a DMRS corresponds to two complex propositions joined by a conjunction, it can be split into two subgraphs corresponding to the coordinated propositions. Each of these subgraphs can then be treated separately by a realization system at a lower cost than the full graph. The subgraph results can then be recombined, yielding the same outcome as performing the operation on the original graph. Apart from conjunctions, chunking can also be applied to subordinated clauses, phrasal complements and to other grammar structures, depending on the demands of the application.

5. Sample Applications

These are included in the pydmrs repository to give some guidance to developers about possible uses of ERS.

5.1. A Simple Robot Interface

In certain applications, the full expressivity of DMRS is not necessary, as we are dealing with a restricted domain. Using a wide-coverage parser like the ERG and post-processing the DMRS output allows us to quickly build a system that can handle a wide range of natural language input, requiring much less developer time than building a system from scratch, while avoiding the need for substantial training data. One such situation is giving commands to a robot, where the range of possible actions is limited. We can first simplify the DMRS (as explained in §4.2) to strip out details that are not relevant in this context. We can then apply mapping rules (as explained in §4.4), so that

commands which should be interpreted in the same way are mapped to identical graphs. For example, we might want to treat *"Turn left at a yellow line"* and *"On a yellow line, turn to the left"* in the same way. The simplified graphs can then be converted to a domain-specific representation, such as low-level instructions for the robot.

5.2. Paraphrase

One application of DMRS mapping is paraphrasing. In terms of DMRS, paraphrases correspond to sets of subgraphs that are considered to be semantically equivalent (in a specific context). Mapping one subgraph to another within a DMRS corresponds to replacing one expression with another in a sentence. We are currently developing a DMRS graph description language as part of the pydmrs library which will allow for a convenient way of specifying such subgraphs. The pydmrs library contains some exemplary paraphrase rules, based on our DMRS mapping functionality with underspecified nodes and written in a preliminary version of the aforementioned description language. In figure 5, we present a few examples of paraphrase rules together with an input sentence and corresponding realization after applying the paraphrase rule.

5.3. Ontological Relationship Extraction

DMRS matching can be used to query subgraph patterns in corpora of parsed text. Searching for DMRS subgraphs is a more robust query method than one based on e.g., regular expressions, since DMRS graphs comprise semantic dependencies and abstract certain purely syntactic phenomena (like active/passive). The effectiveness of a *MRS strategy for extraction of ontological relationships was previously demonstrated by Herbelot and Copestake (2006) using the RMRS representation on Wikipedia. This allowed examples such as the following to be processed:

> The Cottontop Tamarin (Saguinus oedipus), also known as the Pinchu Tamarin, is a small New World monkey weighing less than 1lb (0.5 kg): cottontop tamarin is-a new world monkey

Our current use of DMRS in combination with the Wikipedia dump of WikiWoods (see §3.3) considerably simplifies the work required to achieve such results. The following example illustrates the extraction of the relationship *"X eat Y"* from a few sentences:

A mouse ate the whole cheese. → (mouse, cheese)
Lions eat around 15 zebras per year. → (lion, zebra)
Their children eat so many sweets. → (child, sweet)
Potatoes are mostly eaten by humans. → (human, potato)

Such relationships can be easily expressed in terms of DMRS subgraphs using our DMRS description language.

6. Conclusion

We have outlined a diverse range of applications of ERS, and discussed the need to make it easier to use. We have described why DMRS is preferable to MRS/RMRS for some applications, illustrated some standard types of operation on DMRS and discussed some uses which we are currently making of the representation. We have outlined the DMRS-related resources which we have developed to support developers. The pydmrs repository is currently under active development and we intend to expand it further to support other operations and illustrative applications.

7. Acknowledgements

This research was supported in part by the Singapore MOE Tier 2 grant *That's what you meant: a Rich Representation for Manipulation of Meaning* (MOE ARC41/13) and by PhD studentships funded by Qualcomm, the Schiff Foundation, Churchill College and by the EPSRC (Doctoral Training Grant). We are grateful to the anonymous reviewers for their comments.

8. Bibliographical References

Allen, J. F., Swift, M., and de Beaumont, W. (2008). Deep Semantic Analysis of Text. In Johan Bos et al., editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 343–354. College Publications.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, page 178 – 186, Sofia, Bulgaria.

Bender, E. M., Flickinger, D., Oepen, S., Packard, W., and Copestake, A. (2015). Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 239–249, London, UK.

Bond, F., Oepen, S., Nichols, E., Flickinger, D., Velldal, E., and Haugereid, P. (2011). Deep open-source machine translation. *Machine Translation*, 25:87–105.

Cai, S. and Knight, K. (2013). Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 748–752.

Copestake, A., Flickinger, D., Pollard, C., and Sag, I. A. (2005). Minimal Recursion Semantics. An introduction. *Research on Language and Computation*, 3(4):281–332.

Copestake, A., Corbett, P., Murray-Rust, P., Rupp, C., Siddharthan, A., Teufel, S., and Waldron, B. (2006). An architecture for language processing for scientific texts. *Proceedings of the UK eScience All Hands Meeting 2006*.

Copestake, A. (2007a). Applying robust semantics. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING)*, pages 1–12.

Copestake, A. (2007b). Semantic composition with (robust) minimal recursion semantics. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 73–80, Prague, Czech Republic.

Copestake, A. (2009). Slacker semantics. Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Meeting of*

the European Chapter of the Association for Computational Linguistics, pages 1–9, Athens, Greece.

Dridan, R. and Oepen, S. (2011). Parser evaluation using elementary dependency matching. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 225–230, Dublin, Ireland.

Dridan, R. and Oepen, S. (2013). Document parsing. Towards realistic syntactic analysis. In *Proceedings of the 13th International Conference on Parsing Technologies*, Nara, Japan.

Emerson, G. and Copestake, A. (2015). Leveraging a semantically annotated corpus to disambiguate prepositional phrase attachment. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 1–11, London, UK.

Flickinger, D., Lønning, J. T., Dyvik, H., Oepen, S., and Bond, F. (2005). SEM-I rational MT — enriching deep grammars with a semantic interface for scalable machine translation. In *Proc. MT Summit X*, Phuket, Thailand.

Flickinger, D., Oepen, S., and YtrestÃ¸l, G. (2010). Wikiwoods: Syntacto-semantic annotation for english wikipedia. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

Flickinger, D., Bender, E. M., and Oepen, S. (2014). Towards an encyclopedia of compositional semantics. Documenting the interface of the English Resource Grammar. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 875–881, Reykjavik, Iceland.

Flickinger, D. (2000). On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Frank, A., Krieger, H., Xu, F., Uszkoreit, H., Crysmann, B., Jorg, B., and Schafer, U. (2005). Querying structured knowledge sources. *AAAI-05 Workshop on Question Answering in Restricted Domains*, pages 10–19.

Herbelot, A. and Copestake, A. (2006). Acquiring Ontological Relationships from Wikipedia Using RMRS. In *Proceedings of the ISWC 2006 Workshop on Web Content*.

Herbelot, A. (2013). What is in a text, what isn't, and what this has to do with lexical semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*, pages 321–327, Potsdam, Germany.

Horvat, M., Copestake, A., and Byrne, B. (2015). Hierarchical statistical semantic realization for Minimal Recursion Semantics. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 107–117, London, UK.

Ivanova, A., Oepen, S., Øvrelid, L., and Flickinger, D. (2012). Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 2–11, Jeju, Republic of Korea.

Lien, E. and Kouylekov, M. (2014). Entailment recognition using Minimal Recursion Semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014*, pages 699–703.

MacKinlay, A., Martinez, D., and Baldwin, T. (2009). Biomedical Event Annotation with CRFs and Precision Grammars. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 77–85.

MacKinlay, A., Dridan, R., Flickinger, D., and Baldwin, T. (2010). Cross-domain effects on parse selection for precision grammars. *Research on Language and Computation*, 8(4):299–340.

MacKinlay, A., Martinez, D., and Baldwin, T. (2012). Detecting modification of biomedical events using a deep parsing approach. *BMC medical informatics and decision making*, 12(Suppl 1):S4.

MacKinlay, A. (2012). *Pushing the boundaries of deep parsing*. Ph.D. thesis, Department of Computing and Information Systems, The University of Melbourne.

Oepen, S. and Lønning, J. T. (2006). Discriminant-based MRS banking. In *Proc. LREC-2006*, Genoa, Italy.

Oepen, S., Dyvik, H., Lønning, J. T., Velldal, E., Beermann, D., Carroll, J., Flickinger, D., Hellan, L., Johannessen, J. B., Meurer, P., Nordgård, T., and Rosén, V. (2004). Som å kapp-ete med trollet? Towards MRS-based Norwegian-English machine translation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, Baltimore, MD.

Packard, W., Bender, E. M., Read, J., Oepen, S., and Dridan, R. (2014). Simple negation scope resolution through deep parsing: A semantic solution to a semantic problem. In *Proceedings of the 52nd Meeting of the Association for Computational Linguistics*, pages 69–78, Baltimore, MD, USA.

Packard, W. (2014). UW-MRS: Leveraging a deep grammar for robotic spatial commands. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 812–816, Dublin, Ireland.

Reiplinger, M., Schäfer, U., and Wolska, M. (2012). Extracting glossary sentences from scholarly articles: A comparative evaluation of pattern bootstrapping and deep analysis. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, pages 55–65, Jeju Island, Korea.

Schäfer, U., Kiefer, B., Spurk, C., Steffen, J., and Wang, R. (2011). The ACL Anthology Searchbench. In *Proceedings of the ACL-HLT 2011 System Demonstrations*, pages 7–13, Portland, Oregon, June. Association for Computational Linguistics.

Socher, R., Manning, C. D., and Ng, A. Y. (2010). Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

Yao, X. and Zhang, Y. (2010). Question generation with Minimal Recursion Semantics. In *QG2010: The Third Workshop on Question Generation*.

Yao, X., Bouma, G., and Zhang, Y. (2012). Semantics-based question generation and implementation. *Dialogue & Discourse*, 3(2):11–42.