Encoding Lexicalized Tree Adjoining Grammars with a Nonmonotonic Inheritance Hierarchy

Roger Evans
Information Technology Research Institute
University of Brighton
rpe@itri.bton.ac.uk

Gerald Gazdar
School of Cognitive & Computing Sciences
University of Sussex
geraldg@cogs.susx.ac.uk

David Weir
School of Cognitive & Computing Sciences
University of Sussex
davidw@cogs.susx.ac.uk

Abstract

This paper shows how DATR, a widely used formal language for lexical knowledge representation, can be used to define an LTAG lexicon as an inheritance hierarchy with internal lexical rules. A bottom-up featural encoding is used for LTAG trees and this allows lexical rules to be implemented as covariation constraints within feature structures. Such an approach eliminates the considerable redundancy otherwise associated with an LTAG lexicon.

1 Introduction

The Tree Adjoining Grammar (TAG) formalism was first introduced two decades ago (Joshi et al., 1975), and since then there has been a steady stream of theoretical work using the formalism. But it is only more recently that grammars of non-trivial size have been developed: Abeille, Bishop, Cote & Schabes (1990) describe a feature-based Lexicalized Tree Adjoining Grammar (LTAG) for English which subsequently became the basis for the grammar used in the XTAG system, a wide-coverage LTAG parser (Doran et al., 1994b; Doran et al., 1994a; XTAG Research Group, 1995). The advent of such large grammars gives rise to questions of efficient representation, and the fully lexicalized character of the LTAG formalism suggests that recent research into lexical representation might be a place to look for answers (see for example Briscoe *et al.*(1993); Daelemans & Gazdar(1992)). In this paper we explore this suggestion by showing how the lexical knowledge representation language (LKRL) DATR (Evans & Gazdar, 1989a; Evans & Gazdar, 1989b) can be used to formulate a compact, hierarchical encoding of an LTAG.

The issue of efficient representation for LTAG[1] is discussed by Vijay-Shanker & Schabes (1992), who draw attention to the considerable redundancy inherent in LTAG lexicons that are expressed in a flat manner with no sharing of structure or properties across the elementary trees. For example, XTAG currently includes over 100,000 lexemes, each of which is associated with a family of trees (typically around 20) drawn from a set of over 500 elementary trees. Many of these trees have structure in common, many of the lexemes have the same tree families, and many of the trees within families are systematically related in ways which other formalisms capture using transformations or metarules. However, the LTAG formalism itself does not provide any direct support for capturing such regularities.

Vijay-Shanker & Schabes address this problem by introducing a hierarchical lexicon structure with monotonic inheritance and lexical rules, using an approach loosely based on that of Flickinger (1987) but tailored for LTAG trees rather than HPSG subcategorization lists. Becker (1993; 1994) proposes a slightly different solution, combining an inheritance component and a set of metarules[2]. We share their perception of the problem and agree that adopting a hierarchical approach provides the best available solution to it. However, rather than creating a hierarchical lexical formalism that is specific to the LTAG problem, we have used DATR, an LKRL that is already quite widely known and used. From an LTAG perspective, it makes sense to use an already available LKRL that was specifically designed to address these kinds of representational issues. From a DATR perspective, LTAG presents interesting problems arising from its radically lexicalist character: all grammatical relations, including unbounded dependency constructions, are represented lexically and are thus open to lexical generalization.

There are also several further benefits to be gained from using an established general purpose LKRL such as DATR. First, it makes it easier to compare the resulting LTAG lexicon with those associated with other types of lexical syntax: there are existing DATR

[1]As with all fully lexicalized grammar formalisms, there is really no conceptual distinction to be drawn in LTAG between the lexicon and the grammar: the grammatical rules are just lexical properties.

[2]See Section 6 for further discussion of these approaches.

lexicon fragments for HPSG, PATR and Word Grammar, among others. Second, DATR is not restricted to syntactic description, so one can take advantage of existing analyses of other levels of lexical description, such as phonology, prosody, morphology, compositional semantics and lexical semantics[3]. Third, one can exploit existing formal and implementation work on the language[4].

2 Representing LTAG trees

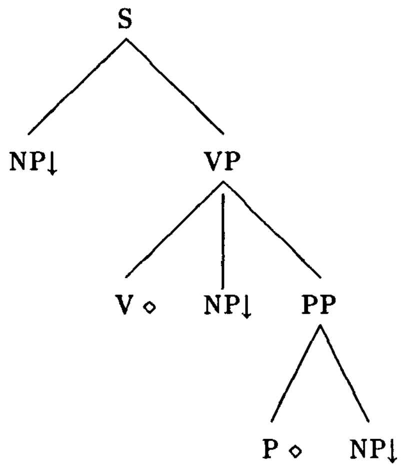

Figure 1: An example LTAG tree for *give*

The principal unit of (syntactic) information associated with an LTAG entry is a tree structure in which the tree nodes are labeled with syntactic categories and feature information and there is at least one leaf node labeled with a **lexical** category (such lexical leaf nodes are known as **anchors**). For example, the canonical tree for a ditransitive verb such as *give* is shown in figure 1. Following LTAG conventions (for the time being), the node labels here are gross syntactic category specifications to which additional featural information may be added[5], and are annotated to indicate node **type**: ◇ indicates an anchor node, and ↓ indicates a substitution node (where a fully specified tree with a compatible root label may be attached)[6].

In representing such a tree in DATR, we do two things. First, in keeping with the radically lexicalist character of LTAG, we describe the tree structure from its (lexical) anchor upwards[7], using a variant of Kilbury's (1990) bottom-up encoding of trees. In this encoding, a tree is described relative to a particular distinguished leaf node (here the anchor node), using binary relations **parent**, **left** and **right**, relating the node to the subtrees associated with its parent, and immediate-left and -right sisters, encoded in the same way. Second, we embed the resulting tree structure (i.e., the node relations and type information) in the feature structure, so that the tree relations (**left**, **right** and **parent**) become features. The obvious analogy here is the use of **first/rest** features to encode subcategorisation lists in frameworks like HPSG.

Thus the syntactic feature information directly associated with the entry for *give* relates to the label for the V node (for example, the value of its **cat** feature is **v**, the value of **type** is **anchor**), while specifications of subfeatures of **parent** relate to the label of the VP node. A simple bottom-up DATR representation for the whole tree (apart from the node type information) follows:

```
Give:
    <cat> = v
    <parent cat> = vp
    <parent left cat> = np
    <parent parent cat> = s
    <right cat> = np
    <right right cat> = p
    <right right parent cat> = pp
    <right right right cat> = np.
```

This says that **Give** is a verb, with VP as its parent, an S as its grandparent and an NP to the left of its parent. It also has an NP to its right, and a tree rooted in a P to the right of that, with a PP parent and NP right sister. The implied bottom-up tree structure is shown graphically in figure 2. Here the nodes are laid out just as in figure 1, but related via **parent**, **left** and **right** links, rather than the more usual (implicitly ordered) daughter links. Notice in particular that the **right** link from the object noun-phrase node points to the *preposition* node, not its phrasal parent – this whole subtree is itself encoded bottom-up. Nevertheless, the full tree structure is completely and accurately represented by this encoding.

[3]See, for example, Bleiching (1992; 1994), Brown & Hippisley (1994), Corbett & Fraser (1993), Cahill (1990; 1993), Cahill & Evans (1990), Fraser & Corbett (in press), Gibbon (1992), Kilgarriff (1993), Kilgarriff & Gazdar (1995), Reinhard & Gibbon (1991).

[4]See, for example, Andry *et al.* (1992) on compilation, Kilbury *et al.* (1991) on coding DAGs, Duda & Gebhardi (1994) on dynamic querying, Langer (1994) on reverse querying, and Barg (1994), Light (1994), Light *et al.* (1993) and Kilbury *et al.* (1994) on automatic acquisition. And there are at least a dozen different DATR implementations available, on various platforms and programming languages.

[5]In fact, LTAG commonly distinguishes two sets of features at each node (**top** and **bottom**), but for simplicity we shall assume just one set in this paper.

[6]LTAG's other tree-building operation is **adjunction**, which allows a tree-fragment to be spliced into the body of a tree. However, we only need to concern ourselves here with the representation of the trees involved, not with the substitution/adjunction distinction.

[7]The tree in figure 1 has more than one anchor – in such cases it is generally easy to decide which anchor is the most appropriate root for the tree (here, the verb anchor).

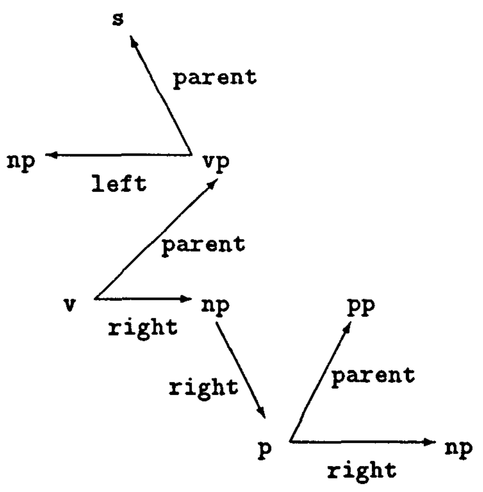

Figure 2: Bottom-up encoding for Give

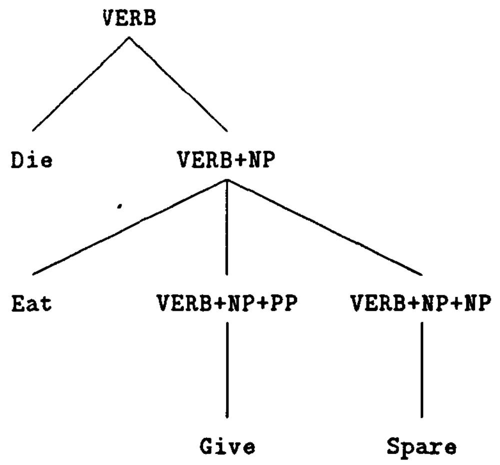

Figure 3: The principal lexical hierarchy

Once we adopt this representational strategy, writing an LTAG lexicon in DATR becomes similar to writing any other type of lexicalist grammar's lexicon in an inheritance-based LKRL. In HPSG, for example, the subcategorisation frames are coded as lists of categories, whilst in LTAG they are coded as trees. But, in both cases, the problem is one of concisely describing feature structures associated with lexical entries and relationships between lexical entries. The same kinds of generalization arise and the same techniques are applicable. Of course, the presence of complete trees and the fully lexicalized approach provide scope for capturing generalizations lexically that are not available to approaches that only identify parent and sibling nodes, say, in the lexical entries.

3 Encoding lexical entries

Following conventional models of lexicon organisation, we would expect Give to have a minimal syntactic specification itself, since syntactically it is a completely regular ditransitive verb. In fact none of the information introduced so far is specific to Give. So rather than providing a completely explicit DATR definition for Give, as we did above, a more plausible account uses an inheritance hierarchy defining abstract intransitive, transitive and ditransitive verbs to support Give (among others), as shown in figure 3.

This basic organisational structure can be expressed as the following DATR fragment[8]:

```
VERB:
    <> == TREENODE
    <cat> == v
    <type> == anchor
    <parent> == VPTREE:<>.

VERB+NP:
    <> == VERB
    <right> == NPCOMP:<>.

VERB+NP+PP:
    <> == VERB+NP
    <right right> == PTREE:<>
    <right right root> == to.

VERB+NP+NP:
    <> == VERB+NP
    <right right> == NPCOMP:<>.

Die:
    <> == VERB
    <root> == die.

Eat:
    <> == VERB+NP
    <root> == eat.

Give:
    <> == VERB+NP+PP
    <root> == give.

Spare:
    <> == VERB+NP+NP
    <root> == spare.
```

Ignoring for the moment the references to TREENODE, VPTREE, NPCOMP and PTREE (which we shall define shortly), we see that VERB defines basic features for all verb entries (and can be used directly for intransitives such as Die), VERB+NP inherits from VERB but adds an NP complement to the right of the verb (for transitives), VERB+NP+PP inherits from VERB+NP but adds a further PP complement and so

[8]To gain the intuitive sense of this fragment, read a line such as <> == VERB as "inherit everything from the definition of VERB", and a line such as <parent> == PPTREE:<> as "inherit the parent subtree from the definition of PPTREE". Inheritance in DATR is always by default – locally defined feature specifications take priority over inherited ones.

on. Entries for regular verb lexemes are then minimal – syntactically they just inherit **everything** from the abstract definitions.

This DATR fragment is incomplete, because it neglects to define the internal structure of the TREENODE and the various subtree nodes in the lexical hierarchy. Each such node is a description of an LTAG tree at some degree of abstraction[9]. The following DATR statements complete the fragment, by providing definitions for this internal structure:

```
TREENODE:
    <> == undef
    <type> == internal.

STREE:
    <> == TREENODE
    <cat> == s.

VPTREE:
    <> == TREENODE
    <cat> == vp
    <parent> == STREE:<>
    <left> == NPCOMP:<>.

NPCOMP:
    <> == TREENODE
    <cat> == np
    <type> == substitution.

PPTREE:
    <> == TREENODE
    <cat> == pp.

PTREE:
    <> == TREENODE
    <cat> ==  p
    <type> == anchor
    <parent> == PPTREE:<>
```

Here, TREENODE represents an abstract node in an LTAG tree and provides a (default) type of internal. Notice that VERB is itself a TREENODE (but with the nondefault type anchor), and the other definitions here define the remaining tree nodes that arise in our small lexicon: VPTREE is the node for VERB's parent, STREE for VERB's grandparent, NPCOMP defines the structure needed for NP complement substitution nodes, etc.[10]

Taken together, these definitions provide a specification for Give just as we had it before, but with the addition of type and root features. They also support some other verbs too, and it should be clear that the basic technique extends readily to a wide range of other verbs and other parts of speech. Also, although the trees we have described are all **initial** trees (in LTAG terminology), we can describe **auxiliary** trees, which include a leaf node of type foot just as easily. A simple example is provided by the following definition for auxiliary verbs:

```
AUXVERB:
    <> == TREENODE
    <cat> == v
    <type> == anchor
    <parent cat> == vp
    <right cat> == vp
    <right type> == foot.
```

4 Lexical rules

Having established a basic structure for our LTAG lexicon, we now turn our attention towards capturing other kinds of relationship among trees. We noted above that lexical entries are actually associated with **tree families**, and that these group together trees that are related to each other. Thus in the same family as a standard ditransitive verb, we might find the full passive, the agentless passive, the dative alternation, the various relative clauses, and so forth. It is clear that these families correspond closely to the outputs of transformations or metarules in other frameworks, but the XTAG system currently has no formal component for describing the relationships among families nor mechanisms for generating them. And so far we have said nothing about them either – we have only characterized single trees.

However, LTAG's large domain of locality means that all such relationships can be viewed as directly lexical, and thus expressible by lexical rules. In fact we can go further than this: because we have embedded the domain of these lexical rules, namely the LTAG tree structures, within the feature structures, we can view such lexical rules as covariation constraints within feature structures, in much the same way that the covariation of, say, syntactic and morphological form is treated. In particular, we can use the mechanisms that DATR already provides for feature covariation, rather than having to invoke in addition some special purpose lexical rule machinery.

We consider six construction types found in the XTAG grammar: passive, dative, subject-auxiliary inversion, *wh*-questions, relative clauses and topicalisation. Our basic approach to each of these is the same. Lexical rules are specified by defining a derived **output** tree structure in terms of an **input** tree structure, where each of these structures is a set of feature specifications of the sort defined above. Each lexical rule has a name, and the input and output tree structures for rule foo are referenced by prefixing feature paths of the sort given above with <input foo ..> or <output foo ..>. So for example, the category of the parent tree node of the output of the passive rule might be referenced as <output passive parent cat>. We define a very general default, stating that the **output** is the same

[9]Even the lexeme nodes are abstract – individual word forms might be represented by further more specific nodes attached below the lexemes in the hierarchy.

[10]Our example makes much use of multiple inheritance (thus, for example, VPTREE inherits from TREENODE, STREE and NPCOMP) but all such multiple inheritance is orthogonal in DATR: no path can inherit from more than one node.

as the input, so that lexical relationships need only concern themselves with components they modify. This approach to formulating lexical rules in DATR is quite general and in no way restricted to LTAG: it can be readily adapted for application in the context of any feature-based lexicalist grammar formalism.

Using this approach, the dative lexical rule can be given a minimalist implementation by the addition of the following single line to VERB+NP+PP, defined above.

```
VERB+NP+PP:
    <output dative right right> == NPCOMP:<>.
```

This causes the second complement to a ditransitive verb in the dative alternation to be an NP, rather than a PP as in the unmodified case. Subject-auxiliary inversion can be achieved similarly by just specifying the output tree structure without reference to the input structure (note the addition here of a form feature specifying verb form):

```
AUXVERB:
    <output auxinv form> == finite-inv
    <output auxinv parent cat> == s
    <output auxinv right cat> == s.
```

Passive is slightly more complex, in that it has to modify the given input tree structure rather than simply overwriting part of it. The definitions for passive occur at the VERB+NP node, since by default, any transitive or subclass of transitive has a passive form. Individual transitive verbs, or whole subclasses, can override this default, leaving their passive tree structure undefined if required. For agentless passives, the necessary additions to the VERB+NP node are as follows[11]:

```
VERB+NP:
    <output passive form> == passive
    <output passive right> ==
            "<input passive right right>".
```

Here, the first line stipulates the form of the verb in the output tree to be passive, while the second line redefines the complement structure: the output of passive has as its first complement the second complement of its input, thereby discarding the first complement of its input. Since complements are daisy-chained, all the others move up too.

Wh-questions, relative clauses and topicalisation are slightly different, in that the application of the lexical rule causes structure to be added to the top of the tree (above the S node). Although these constructions involve unbounded dependencies, the unboundedness is taken care of by the LTAG adjunction mechanism: for lexical purposes the dependency is local. Since the relevant lexical rules can apply to sentences that contain any kind of verb, they need to be stated at the VERB node. Thus, for example, topicalisation and *wh*-questions can be defined as follows:

[11]Oversimplifying slightly, the double quotes in "<input passive right right>" mean that that DATR path will not be evaluated locally (i.e., at the VERB+NP node), but rather at the relevant lexeme node (e.g., Eat or Give).

```
VERB:
    <output topic parent parent parent cat>
                                          == s
    <output topic parent parent left cat> == np
    <output topic parent parent left form>
                                     == normal
    <output whq> == "<output topic>"
    <output whq parent parent left form> == wh.
```

Here an additional NP and S are attached above the original S node to create a topicalised structure. The *wh*-rule inherits from the topicalisation rule, changing just one thing: the form of the new NP is marked as wh, rather than as normal. In the full fragment[12], the NP added by these rules is also syntactically cross-referenced to a specific NP marked as null in the input tree. However, space does not permit presentation or discussion of the DATR code that achieves this here.

5 Applying lexical rules

As explained above, each lexical rule is defined to operate on its own notion of an input and produce its own output. In order for the rules to have an effect, the various input and output paths have to be linked together using inheritance, creating a chain of inheritances between the base, that is, the canonical definitions we introduced in section 3, and surface tree structures of the lexical entry. For example, to 'apply' the dative rule to our Give definition, we could construct a definition such as this:

```
Give-dat:
    <> == Give
    <input dative> == <>
    <surface> == <output dative>.
```

Values for paths prefixed with surface inherit from the output of the dative rule. The input of the dative rule inherits from the base (unprefixed) case, which inherits from Give. The dative rule definition (just the one line introduced above, plus the default that output inherits from input) thus mediates between Give and the surface of Give-dat. This chain can be extended by inserting additional inheritance specifications (such as passive). Note that surface defaults to the base case, so all entries have a surface defined.

However, in our full fragment, additional support is provided to achieve and constrain this rule chaining. Word definitions include boolean features indicating which rules to apply, and the presence of these features trigger inheritance between appropriate input and output paths and the base and surface specifications at the ends of the chain. For example, Word1 is an alternative way of specifying the dative alternant of Give, but results in inheritance linking equivalent to that found in Give-dat above:

[12]The full version of this DATR fragment includes all the components discussed above in a single coherent, but slightly more complex account. It is available on request from the authors.

```
Word1:
    <> == Give
    <alt dative> == true.
```

More interestingly, Word2 properly describes a *wh*-question based on the agentless passive of the dative of Give.

```
Word2:
    <> == Give
    <alt whq> == true
    <alt dative> == true
    <alt passive> == true.
    <parent left form> == null
```

Notice here the final line of Word2 which specifies the location of the 'extracted' NP (the subject, in this case), by marking it as null. As noted above, the full version of the whq lexical rule uses this to specify a cross-reference relationship between the *wh*-NP and the null NP.

We can, if we wish, encode constraints on the applicability of rules in the mapping from boolean flags to actual inheritance specifications. Thus, for example, whq, rel, and topic are mutually exclusive. If such constraints are violated, then no value for surface gets defined. Thus Word3 improperly attempts topicalisation in addition to *wh*-question formation, and, as a result, will fail to define a surface tree structure at all:

```
Word3:
    <> == Give
    <alt whq> == true
    <alt topic> == true
    <alt dative> == true
    <alt passive> == true
    <parent left form> == null.
```

This approach to lexical rules allows them to be specified at the appropriate point in the lexical hierarchy, but overridden or modified in subclasses or lexemes as appropriate. It also allows default generalisation over the lexical rules themselves, and control over their application. The last section showed how the whq lexical rule could be built by a single minor addition to that for topicalisation. However, it is worth noting that, in common with other DATR specifications, the lexical rules presented here are **rule instances** which can only be applied once to any given lexeme – multiple application could be supported, by making multiple instances inherit from some common rule specification, but in our current treatment such instances would require different rule names.

6 Comparison with related work

As noted above, Vijay-Shanker & Schabes (1992) have also proposed an inheritance-based approach to this problem. They use monotonic inheritance to build up partial descriptions of trees: each description is a finite set of dominance, immediate dominance and linear precedence statements about tree nodes in a tree description language developed by Rogers & Vijay-Shanker (1992), and category information is located in the node labels.

This differs from our approach in a number of ways. First, our use of nonmonotonic inheritance allows us to manipulate total instead of partial descriptions of trees. The abstract verb class in the Vijay-Shanker & Schabes account subsumes both intransitive and transitive verb classes but is not identical to either – a minimal-satisfying-model step is required to map partial tree descriptions into actual trees. In our analysis, VERB is the intransitive verb class, with complements specifically marked as undefined: thus VERB:<right> == undef is inherited from TREENODE and VERB+NP just overrides this complement specification to add an NP complement. Second, we describe trees using only local tree relations (between adjacent nodes in the tree), while Vijay-Shanker & Schabes also use a nonlocal dominance relation.

Both these properties are crucial to our embedding of the tree structure in the feature structure. We want the category information at each tree node to be partial in the conventional sense, so that in actual use such categories can be extended (by unification or whatever). So the feature structures that we associate with lexical entries must be viewed as partial. But we do not want the tree structure to be extendible in the same way: we do not want an intransitive verb to be applicable in a transitive context, by unifying in a complement NP. So the tree structures we define must be total descriptions[13]. And of course, our use of only local relations allows a direct mapping from tree structure to feature path, which would not be possible at all if nonlocal relations were present.

So while these differences may seem small, they allow us to take this significant representational step – significant because it is the tree structure embedding that allows us to view lexical rules as feature covariation constraints. The result is that while Vijay-Shanker & Schabes use a tree description language, a category description language and a further formalism for lexical rules, we can capture everything in one framework all of whose components (nonmonotonicity, covariation constraint handling, etc.) have already been independently motivated for other aspects of lexical description[14].

Becker's recent work (1993; 1994) is also directed at exactly the problem we address in the present paper. Like him, we have employed an inheritance hierarchy. And, like him, we have employed a set of lexical rules (corresponding to his metarules). The key differences between our account and his are (i)

[13]Note that simplified fragment presented here does not get this right. It makes all feature specifications total descriptions. To correct this we would need to change TREENODE so that only the values of <right>, <left> and <parent> default to undef.

[14]As in the work cited in footnote 3, above.

that we have been able to use an existing lexical knowledge representation language, rather than designing a formal system that is specific to LTAG, and (ii) that we have expressed our lexical rules in exactly the same language as that we have used to define the hierarchy, rather than invoking two quite different formal systems.

Becker's sharp distinction between his metarules and his hierarchy gives rise to some problems that our approach avoids. Firstly, he notes that his metarules are subject to lexical exceptions and proposes to deal with these by stating "for each entry in the (syntactic) lexicon .. which metarules are applicable for this entry" (1993,126). We have no need to carry over this use of (meta)rule features since, in our account, lexical rules are not distinct from any other kind of property in the inheritance hierarchy. They can be stated at the most inclusive relevant node and can then be overridden at the exceptional descendant nodes. Nothing specific needs to be said about the nonexceptional nodes.

Secondly, his metarules may themselves be more or less similar to each other and he suggests (1994,11) that these similarities could be captured if the metarules were also to be organized in a hierarchy. However, our approach allows us to deal with any such similarities in the main lexical hierarchy itself[15] rather than by setting up a separate hierarchical component just for metarules (which appears to be what Becker has in mind).

Thirdly, as he himself notes (1993,128), because his metarules map from elementary trees that are in the inheritance hierarchy to elementary trees that are outside it, most of the elementary trees actually used are not directly connected to the hierarchy (although their derived status with respect to it can be reconstructed). Our approach keeps all elementary trees, whether or not they have been partly defined by a lexical rule, entirely within the lexical hierarchy.

In fact, Becker himself considers the possibility of capturing all the significant generalizations by using just one of the two mechanisms that he proposes: "one might want to reconsider the usage of one mechanism for phenomena in both dimensions" (1993,135). But, as he goes on to point out, his existing type of inheritance network is not up to taking on the task performed by his metarules because the former is monotonic whilst his metarules are not. However, he does suggest a way in which the hierarchy could be completely replaced by metarules but argues against adopting it (1993,136).

As will be apparent from the earlier sections of this paper, we believe that Becker's insights about the organization of an LTAG lexicon can be better expressed if the metarule component is replaced by an encoding of (largely equivalent) lexical rules that are an integral part of a nonmonotonic inheritance hierarchy that stands as a description of all the elementary trees.

[15]As illustrated by the way in which the whq lexical rule inherits from that for topicalisation in the example given above.

Acknowledgements

A precursor of this paper was presented at the September 1994 TAG+ Workshop in Paris. We thank the referees for that event and the ACL-95 referees for a number of helpful comments. We are also grateful to Aravind Joshi, Bill Keller, Owen Rambow K. Vijay-Shanker and The XTAG Group. This research was partly supported by grants to Evans from SERC/EPSRC (UK) and to Gazdar from ESRC (UK).

References

Anne Abeille, Kathleen Bishop, Sharon Cote, & Yves Schabes. 1990. A lexicalized tree adjoining grammar for english. Technical Report MS-CIS-90-24, Department of Computer & Information Science, Univ. of Pennsylvania.

Francois Andry, Norman Fraser, Scott McGlashan, Simon Thornton, & Nick Youd. 1992. Making DATR work for speech: lexicon compilation in SUNDIA. *Comput. Ling.*, 18(3):245–267.

Petra Barg. 1994. Automatic acquisition of datr theories from observations. Theories des lexicons: Arbeiten des sonderforschungsbereichs 282, Heinrich-Heine Univ. of Duesseldorf, Duesseldorf.

Tilman Becker. 1993. *HyTAG: A new type of Tree Adjoining Grammar for hybrid syntactic representation of free word order languages.* Ph.D. thesis, Univ. des Saarlandes.

Tilman Becker. 1994. Patterns in metarules. In *Proceedings of the Third International Workshop on Tree Adjoining Grammars*, 9–11.

Doris Bleiching. 1992. Prosodisches wissen in lexicon. In G. Goerz, ed., *KONVENS-92*, 59–68. Springer-Verlag.

Doris Bleiching. 1994. Integration von morphophonologie und prosodie in ein hierarchisches lexicon. In H. Trost, ed., *Proceedings of KONVENS-94*, 32–41.

Ted Briscoe, Valeria de Paiva, & Ann Copestake. 1993. *Inheritance, Defaults, & the Lexicon.* CUP.

Dunstan Brown & Andrew Hippisley. 1994. Conflict in russian genitive plural assignment: A solution represented in DATR. *J. of Slavic Linguistics*, 2(1):48–76.

Lynne Cahill & Roger Evans. 1990. An application of DATR: the TIC lexicon. In *ECAI-90*, 120–125.

Lynne Cahill. 1990. Syllable-based morphology. In *COLING-90*, volume 3, 48–53.

Lynne Cahill. 1993. Morphonology in the lexicon. In *EACL-93*, 37–96.

Greville Corbett & Norman Fraser. 1993. Network morphology: a DATR account of Russian nominal inflection. *J. of Linguistics*, 29:113–142.

Walter Daelemans & Gerald Gazdar, eds. 1992. Special issues on inheritance. *Comput. Ling.*, 18(2 & 3).

Christy Doran, Dania Egedi, Beth Ann Hockey, & B. Srinivas. 1994a. Status of the XTAG system. In *Proceedings of the Third International Workshop on Tree Adjoining Grammars*, 20–23.

Christy Doran, Dania Egedi, Beth Ann Hockey, B. Srinivas, & Martin Zaidel. 1994b. XTAG system — a wide coverage grammar for english. In *COLING-94*, 922–928.

Markus Duda & Gunter Gebhardi. 1994. DUTR – a DATR-PATR interface formalism. In H. Trost, ed., *Proceedings of KONVENS-94*, 411–414.

Roger Evans & Gerald Gazdar. 1989a. Inference in DATR. In *EACL-89*, 66–71.

Roger Evans & Gerald Gazdar. 1989b. The semantics of DATR. In *AISB-89*, 79–87.

Daniel P. Flickinger. 1987. *Lexical Rules in the Hierarchical Lexicon*. Ph.D. thesis, Stanford Univ.

Norman Fraser & Greville Corbett. in press. Gender, animacy, & declensional class assignment: a unified account for russian. In Geert Booij & Jaap van Marle, ed., *Yearbook of Morphology 1994*. Kluwer, Dordrecht.

Dafydd Gibbon. 1992. ILEX: a linguistic approach to computational lexica. In Ursula Klenk, ed., *Computatio Linguae: Aufsa("tze zur algorithmischen und quantitativen Analyse der Sprache (Zeitschrift fu("r Dialektologie und Linguistik, Beiheft 73)*, 32–53. Franz Steiner Verlag, Stuttgart.

A. K. Joshi, L. S. Levy, & M. Takahashi. 1975. Tree adjunct grammars. *J. Comput. Syst. Sci.*, 10(1):136–163.

James Kilbury, Petra [Barg] Naerger, & Ingrid Renz. 1991. DATR as a lexical component for PATR. In *EACL-91*, 137–142.

James Kilbury, Petra Barg, & Ingrid Renz. 1994. Simulation lexicalischen erwerbs. In Christopher Habel & Gert Rickheit Sascha W. Felix, ed, *Kognitive Linguistik: Repraesentation und Prozesse*, 251–271. Westdeutscher Verlag, Opladen.

James Kilbury. 1990. Encoding constituent structure in feature structures. Unpublished manuscript, Univ. of Duesseldorf, Duesseldorf.

Adam Kilgarriff & Gerald Gazdar. 1995. Polysemous relations. In Frank Palmer, ed., *Grammar & meaning: essays in honour of Sir John Lyons*, 1–25. CUP.

Adam Kilgarriff. 1993. Inheriting verb alternations. In *EACL-93*, 213–221.

Hagen Langer. 1994. Reverse queries in DATR. In *COLING-94*, 1089–1095.

Marc Light, Sabine Reinhard, & Marie Boyle-Hinrichs. 1993. INSYST: an automatic inserter system for hierarchical lexica. In *EACL-93*, page 471.

Marc Light. 1994. Classification in feature-based default inheritance hierarchies. In H. Trost, ed., *Proceedings of KONVENS-94*, 220–229.

Sabine Reinhard & Dafydd Gibbon. 1991. Prosodic inheritance & morphological generalisations. In *EACL-91*, 131–136.

James Rogers & K. Vijay-Shanker. 1992. Reasoning .with descriptions of trees. In *ACL-92*, 72–80.

K. Vijay-Shanker & Yves Schabes. 1992. Structure sharing in lexicalized tree-adjoining grammar. In *COLING-92*, 205–211.

The XTAG Research Group. 1995. A lexicalized tree adjoining grammar for English. Technical Report IRCS Report 95-03, The Institute for Research in Cognitive Science, Univ. of Pennsylvania.