Proceedings of EACL '99

Representing a System of Lexical Types Using Default Unification

Aline Villavicencio
Computer Laboratory
University of Cambridge
New Museums Site
Pembroke Street
Cambridge CB2 3QG
ENGLAND
Aline.Villavicencio@cl.cam.ac.uk

Abstract

Default inheritance is a useful tool for encoding linguistic generalisations that have exceptions. In this paper we show how the use of an order independent typed default unification operation can provide non-redundant highly structured and concise representation to specify a network of lexical types, that encodes linguistic information about verbal subcategorisation. The system of lexical types is based on the one proposed by Pollard and Sag (1987), but uses the more expressive typed default feature structures, is more succinct, and able to express linguistic sub-regularities more elegantly.

1 Introduction

Several authors have highlighted the importance of using defaults in the representation of linguistic knowledge, in order to get linguistically adequate descriptions for some natural language phenomena ((Gazdar, 1987), (Bouma, 1992), (Daelemans et al, 1992), (Briscoe, 1993)). Defaults have been used in the definition of inflectional morphology, specification of lexical semantics, analysis of gapping constructions and ellipsis among others. In this paper we use defaults to structure the lexicon, concentrating on the description of verbal subcategorisation information.

The issue of how to organise lexical information is especially important when a lexicalised formalism like Categorial Grammar (CG) or Head-Driven Phrase Structure Grammar (HPSG) is employed, since the burden of linguistic description is concentrated in the lexicon and if lexical entries are organised as unrelated lists, there is a significant loss of generalisation and an increase in redundancy. Alternatively, it is possible to use inheritance networks, which provide representations that are able to capture linguistic regularities about classes of items that behave similarly. This idea is employed in Pollard and Sag's (1987) sketch of an HPSG lexicon as a monotonic multiple orthogonal inheritance type hierarchy. However, this work fail to make use of defaults, which would significantly reduce redundancy in lexical specifications and would enable them to elegantly express sub-regularities (Krieger and Nerbonne, 1993). In this paper we demonstrate that using default unification, namely the order-independent and persistent version of default unification described in (Lascarides et al, 1996b) and (Lascarides and Copestake, 1999), to implement a default inheritance network results in a fully declarative specification of a lexical fragment based on Pollard and Sag's (1987), but that is both more succinct and able to express elegantly linguistic sub-regularities, such as the marked status of subject control of transitive subject-control verbs.

In section 2, a brief description of the use of defaults and YADU is given. In section 3, we present the results of representing the proposed lexical fragment in terms of default multiple inheritance networks. Finally, we discuss the results achieved and future work.

2 Default Inheritance and YADU

In this work, a default multiple orthogonal inheritance network is used to represent lexical information. Thus, with different subnetworks used to encode different kinds of linguistic knowledge, the idea is that linguistic regularities are encoded near the top of the network, while nodes further down the network are used to represent sub-regularities or exceptions. Such an approach to representing the lexicon has some advantages, like its ability to capture linguistic generalisations, conciseness, uniformity, ease of maintenance and modification, and modularity (Daelemans et al, 1992).

This default multiple inheritance network is implemented using YADU (Lascarides and Copestake, 1999), which is an order independent default unification operation on typed feature structures (TFS). YADU uses an extended definition of TFSs called typed default feature structures (TDFSs), to explicitly distinguish the non-default information from the default one, where a TDFS is composed by an indefeasible TFS (I), which contains the non-default information and a defeasible TFS (D), which contains the default information, with a '/' separating these two TFSs (I on the left-hand and D on the right-hand). As a consequence, during default unification non-default information can always be preserved and only consistent default information is incorporated into the defeasible TFS. Another important point is that default unification of two feature structures is deterministic, always returning a single value. Moreover, default specifications can be made to act as indefeasible information, using YADU's *DefFill* operation (Lascarides and Copestake, 1999), that has a TDFS as input and returns a TFS by incorporating all the default information into the indefeasible TFS, say at the interface between the lexicon and the rest of the system. YADU also provides the possibility of defining defaults that are going to persist outside the lexicon, with the *p* operator (Lascarides et al, 1996b), which was already shown to be significant, for example, for the interface between the lexicon and pragmatics, where lexically encoded semantic defaults can be overridden by discourse information (Lascarides et al, 1996a). Furthermore, YADU supports the definition of inequalities, which are used to override default reentrancies when no conflicting values are defined in the types involved (Lascarides and Copestake, 1999).

YADU ($\overset{<>}{\sqcap}$) can be informally defined as an operation that takes two TDFSs and produces a new one, whose indefeasible part is the result of unifying the indefeasible information defined in the input TDFSs; and the defeasible part is the result of combining the indefeasible part with the maximal set of compatible default elements, according to type specificity, as shown in the example below. Throughout this paper we adopt the abbreviatory notation from (Lascarides et al, 1996b) where *Indefeasible/Defeasible* is abbreviated to *Indefeasible* if *Indefeasible* = *Defeasible* and $\top$/*Defeasible* is abbreviated to /*Defeasible*.

$$t' \sqsubset t$$

$$\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t'} \\ \mathrm{F} : /\mathbf{a} \\ \mathrm{G} : /\mathbf{b} \end{bmatrix}$$

- $I = \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} \sqcap \begin{bmatrix} \mathbf{t'} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} = \begin{bmatrix} \mathbf{t'} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix}$

- $D = \begin{bmatrix} \mathbf{t'} \\ \mathrm{F} : \mathbf{a} \\ \mathrm{G} : \mathbf{b} \end{bmatrix}$

For a more detailed introduction to YADU see (Lascarides and Copestake, 1999).

3 The proposed lexical network

The proposed verbal subcategorisation hierarchy[1], which is based on the sketch by Pollard and Sag (1987) is shown in figure 1. In this hierarchy, types are ordered according to the number and type of the subcategorisation arguments they specify. The subcategorisation arguments of a particular category[2] are defined in its SUBCAT feature as a difference-list. Thus, the verbal hierarchy starts with the *intrans* type, which by default specifies the need for exactly one argument, the NP subject, where *e-list* is a type that marks the end of the subcategorisation list:

(1) *intrans* type:
[SUBCAT: <HEAD: **np**, TAIL: /**e-list**>].

Now all the attributes specified for the subcategorised subject NP in *intrans* are inherited by instances of this type and by its subtypes[3], namely, *trans* and *intrans-control*. However, since these types subcategorise for 2 arguments, they need to override the default of exactly one argument, specified by the *e-list* value for TAIL, and add an extra argument: an NP object for *trans*, and a predicative complement for *intrans-control*. In this way, the specification of the *trans* type is:

(2) *trans* type:
[SUBCAT:<TAIL: HEAD: **np**, TAIL: TAIL: /**e-list**>].

Similarly, the instances and subtypes of *trans* inherit from *intrans* all the attributes for the subject NP and from *trans* the attributes for the object NP, in addition to their own constraints. With the use of defaults there is no need for specifying a type like *strict-trans*, as defined in Pollard and Sag's hierarchy, since it contains exactly the same information as their *trans* type, except that the former specifies the SUBCAT

[1]For reasons of space we are only showing the parts of the lexical hierarchy that are relevant for this paper.

[2]Linguistic information is expressed using a simplified notation for the SUBCAT list, and for reasons of clarity, we are only showing categories in an atomic form, without the attributes defined.

[3]In this paper, we are not assuming the coverage condition, that any type in a hierarchy has to be resolvable to a most specific type.

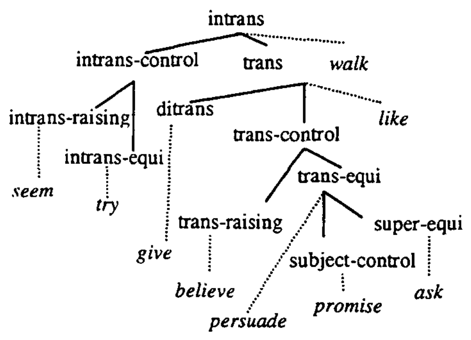

Figure 1: The Proposed Hierarchy

attribute as containing **exactly** two arguments:

(3) Pollard and Sag's *strict-trans* type:
[SUBCAT: <HEAD: **np**, TAIL: HEAD: **np**, TAIL: TAIL: **e-list**>],

while the latter works as an intermediate type, where SUBCAT contains **at least** two arguments, as shown in (4), offering its subtypes the possibility of adding extra arguments.

(4) Pollard and Sag's *trans* type:
[SUBCAT: <HEAD: **np**, TAIL: HEAD: **np**>],

Defaults automatically provide this possibility, by defeasibly marking the end of the subcategorisation list, which defines the number of arguments needed, avoiding the need for these redundant specifications, where the information contained in one lexical sign is repeated in others. Furthermore, these defaults are used to capture lexical generalisations, but outside the lexicon, we want them to act as indefeasible constraints; therefore, we apply the *DefFill* operation to these default specifications, except where marked as persistently default. In this way, a type like *trans*, after *DefFill*, has the consistent defaults incorporated and specifies, indefeasibly the need for exactly two arguments, as Pollard and Sag's *strict-trans* shown in (3):

(5) *trans* type *DefFill*ed:
[SUBCAT: <HEAD: **np**, TAIL: HEAD: **np**, TAIL: TAIL: **e-list**>].

Apart from supporting this kind of generalisation, defaults are also used to express sub-regularities, as, for example, in the case of *super-equi* and *subject-control* verbs, which are both exceptions to the general case specified by *trans-equi*. The type *trans-equi* encodes transitive-equi verbs by specifying that the predicative complement of the transitive verb is by default controlled by the object (e.g. *The teacher persuaded* ***the doctor*** *to go*):

(6) *trans-equi* type:
[SUBCAT: <TAIL: HEAD: **np**/$\boxed{1}$, TAIL: TAIL: HEAD: **vp**(INF, SUBCAT:<HEAD: **np**/$\boxed{1}$ >), TAIL: TAIL: TAIL: **e-list**>].

For super-equi verbs, the predicative complements can be controlled by either the object or the subject. Therefore, the default object-control in the *super-equi* type, inherited from *trans-equi*, should be explicitly marked with the *p* operator to persist until discourse interpretation, as shown in (7), since all other features are made indefeasible prior to parsing.

(7) *super-equi* type:
[SUBCAT: <TAIL: HEAD: **np**$/_p\boxed{1}$, TAIL: TAIL: HEAD: **vp**(INF, SUBCAT: <HEAD: **np**$/_p\boxed{1}$ >) >].

This default would only survive in the absence of conflicting discourse information (as in e.g.: *They needed someone with medical training. So, the teacher asked* ***the doctor*** *to go (since she had none)*, which is object-controlled). Otherwise, if there is conflicting information, this default is rejected (as in e.g.: *They needed someone with teaching experience. So,* ***the teacher*** *asked the doctor (to be allowed) to go*, where the control is by the subject). A description of the precise mechanism to do this can be found in (Lascarides et al, 1996a). Transitive subject-control verbs follow the pattern specified by *trans-equi*, but contrary to this pattern, it is the subject that controls the predicative complement and not the object (e.g. ***The teacher*** *promised to go*):

(8) *subject-control* type:
[SUBCAT: <HEAD: **np**$\boxed{2}$, TAIL: HEAD: **np**/$\boxed{1}$, TAIL: TAIL: HEAD: **vp**(INF, SUBCAT: <HEAD: **np**$\boxed{2}$ >) >, $\boxed{1} \not\leftrightarrow \boxed{2}$].

In this case, the constraint on *subject-control* specifies that the coindexation is determined by the subject, and as it does not conflict with the default coindexation by the object-control, *inequalities* ($\not\leftrightarrow$) are used to remove the default value.

As a result of using default inheritance to represent information about verbal subcategorisation, it is possible to obtain a highly structured and succinct hierarchy. In comparison with the hierarchy defined by Pollard and Sag (1987), this one avoids the need of redundant specifications and associated type declarations, like the *strict-trans* type, which are needed in a monotonic encoding. In this way, while Pollard and Sag's hierarchy is defined using 23 nodes, this is defined using only 19 nodes, and by defining 2 more nodes, it is possible to specify *subject-control* and *super-equi* types. By avoiding this redundancy, there is a real gain in conciseness, with the resulting hierarchy extending the information defined in Pollard and Sag's, with the addition of sub-regularities, in a more compact encoding.

4 Conclusion

In this paper we demonstrated how the use of default unification in the organisation of lexical information can provide non-redundant description of lexical types. In this way, we implemented a default inheritance network that represents verbal subcategorisation information, using YADU. It resulted in a significant reduction in lexical redundancy, with linguistic regularities and subregularities defined by means of TDFSs, in a lexicon that is succinctly organised, and that is also easier to maintain and modify, when compared to its monotonic counterpart. The resulting verbal hierarchy is able not only to encode the same information as Pollard and Sag's but also to specify more sub-regularities, in a more concise way. Such an approach has the advantage of optionally allowing default specifications to persist outside the lexicon, which is important for the specification of control in super-equi verbs and for lexical semantics. Moreover, as an order independent operation, it provides a declarative mechanism for default specification, with no cost in formal elegance. Finally, as YADU operates directly on feature structures, defaults are allowed as a fully integrated part of the typed feature structure system, and, as a consequence YADU integrates well with constraint-based formalisms. Further work will complement these results by comparing the adequacy of different default unification operations, like the one used in DATR, for this kind of linguistic description. This work is part of a larger project concerned with the investigation of grammatical acquisition within constraint-based formalisms.

5 Acknowledgements

I would like to thank Ted Briscoe, Ann Copestake and Fabio Nemetz for their comments and advice on this paper. Thanks also to the anonymous reviewers for their comments. The research reported on this paper is supported by doctoral studentship from CAPES/Brazil.

References

Bouma, Gosse. 1992. Feature Structures and Nonmonotonicity. *Computational Linguistics*, 18.2.

Briscoe, Ted. 1993. Introduction. *Inheritance, Defaults and the Lexicon.* Ted Briscoe, Ann Copestake and Valeria de Paiva eds. Cambridge University Press, Cambridge.

Daelemans, Walter, Koenraad De Smedt and Gerald Gazdar. 1992. Inheritance in Natural Language Processing. *Computational Linguistics*, 18.2.

Gazdar, Gerald. 1987. Linguistic Applications of Default Inheritance Mechanisms. *Linguistic Theory and Computer Applications.* Pete Whitelock, Mary M. Wood, Harold Somers, Rod Johnson and Paul Bennett eds.

Krieger, Hans-Ulrich. and John Nerbonne. 1993. Feature-Based Inheritance Networks for Computational Lexicons. *Inheritance, Defaults and the Lexicon.* Ted Briscoe, Ann Copestake and Valeria de Paiva eds. Cambridge University Press, Cambridge.

Lascarides, Alex, Ann Copestake and Ted Briscoe. 1996a. Ambiguity and Coherence. *Journal of Semantics*, 13.1, 41-65.

Lascarides, Alex, Ted Briscoe, Nicholas Asher and Ann Copestake. 1996b. Order Independent Persistent Typed Default Unification. *Linguistics and Philosophy* , 19.1, 1-89.

Lascarides, Alex and Ann Copestake. 1999. Default Representation in Constraint-based Frameworks. *To appear in Computational Linguistics*, 25.2. An earlier version of the paper is available at http://www.csli.stanford.edu/ ~aac/ papers/ yadu.gz

Pollard, Carl. and Ivan A. Sag. 1987. *Information-Based Syntax and Semantics*, CSLI lecture notes series, Number 13.