Inheritance in lexical representation

Ann Copestake
CSLI, Stanford University,
Stanford, CA 94305, USA
aac@csli.stanford.edu

1 Introduction

In this abstract I give an overview of some recent work on inheritance in lexical representation. I have not attempted to be comprehensive, in particular, I will concentrate on work within unification or constraint based approaches. Since this overview can necessarily only be very sketchy, I have given a fairly extensive bibliography. See in particular, Daelemans *et al* (1992) which gives an extensive review of inheritance in natural language processing, and the collection of papers in Briscoe *et al* (1993).

It is necessary for a theoretically adequate lexical representation language to be able to capture linguistically relevant generalisations about classes of items which behave in similar ways with respect to phonology, morphology, syntax or semantics. Such mechanisms are now standardly assumed in the computational linguistics literature, but for some examples of linguistic works where inheritance is explicitly discussed, see Pollard and Sag, 1987; Flickinger, 1987; Flickinger and Nerbonne, 1992; Sanfilippo, 1993. One of the most important techniques is to specify constraints on classes of lexical item, which will be inherited by subclasses. In some ways this is very similar to techniques from object-oriented programming, and indeed several researchers have explicitly used object-oriented techniques (e.g. Daelemans, 1987). But, in general, the assumption is that special-purpose data structures are appropriate for linguistic representation and that the operations available on these structures should be restricted.

It is sometimes assumed that the lexicon will be compiled off-line: i.e. all the lexical entries will be expanded before parsing takes place. This is, however, not feasible for anything other than restricted domain systems if a lexicalist grammar framework (e.g. LFG, HPSG, categorial grammar) is being assumed. For example, a fully expanded entry in the ACQUILEX LKB system (e.g. Copestake *et al*, 1993) for an intransitive verb occupies about 10K of memory (using a categorial grammar with no detailed lexical semantic information). Even a moderate coverage lexicon would have to contain at least 50,000 such entries (i.e. it would occupy 500 Megabytes).[1] Obviously, data compression techniques could be used to reduce the information stored, but standard compression algorithms will only reduce the size of the entry by a factor of two or so, whereas the use of inheritance can easily reduce it by a factor of 100. Inheritance is therefore relevant from a computational viewpoint, as well as a theoretical one.

The final reason for wanting to capture generalisations as efficiently and perspicuously as possible, is to simplify the task of constructing, developing and maintaining a lexicon. Strict inheritance ensures consistency, e.g. all intransitive verbs have to have the properties specified for that class. More stringent consistency checks are possible, to ensure that a lexical entry does not have any spurious attributes, for example. Default inheritance, on the other hand, allows for exceptions. A default inheritance hierarchy can always be reconstructed as a strict hierarchy, but at a cost of increasing the number of classes and of making the generalisations less obvious. One of the main questions in lexical representation is whether defaults should be used, and if so, how their use can be formalised. I discuss some recent work in §3, after briefly considering monotonic inheritance and typed feature structures.

2 Inheritance and typed feature structures

Most computational linguistic systems have some mechanism for inheritance. Most basically, a device is provided so that a name can be provided as an abbreviation for a set of descriptions: if the descriptions can include other abbreviatory names this gives an equivalent effect to an inheritance hierarchy. I will use the term *template*, as in PATR-II (Shieber, 1986), for this style of inheritance mechanism, though they are also often referred to as *macros*. Templates are part of the language for describing feature structures (FSs) and lexical entries in particular, rather than being themselves part of the feature structure language. The top of Figure 1 shows an example of the use of templates.

One deficiency of languages such as PATR-II was that values were either totally underspecified, in which case they were compatible with any other value, or fully specified, in which case they were only

[1] This estimate is conservative. Learner's dictionaries of English tend to contain around 50,000 headwords, but many of these would be expanded into multiple lexical structures. Furthermore, many grammars have more complex lexical structures than this.

```
Intrans-verb
< SYN CAT > = verb
< SYN SUBJ SYN CAT > = np

Monadic
< SEM PRED > = pred
< SYN SUBJ SEM > = < SEM PRED ARG1 >

Monadic-stative-intrans-verb
Monadic Intrans-verb
< SEM PRED > = stative
```

$$\textbf{intrans-verb} \rightarrow \left[\text{SYN} : \begin{bmatrix} \text{CAT} : \textbf{verb} \\ \text{SUBJ} : \left[\text{SYN} : \left[\text{CAT} : \textbf{np} \right] \right] \end{bmatrix} \right]$$

$$\textbf{monadic} \rightarrow \begin{bmatrix} \text{SYN} : \left[\text{SUBJ} : \left[\text{SEM} : \boxed{1} \right] \right] \\ \text{SEM} : \begin{bmatrix} \text{PRED} : \textbf{pred} \\ \text{ARG1} : \boxed{1} \end{bmatrix} \end{bmatrix}$$

$$\textbf{monadic-stative-intrans-verb} \rightarrow \textbf{monadic} \wedge \textbf{intrans-verb} \wedge \left[\text{SEM} : \left[\text{PRED} : \textbf{stative} \right] \right]$$

Figure 1: Templates and sorts/types

compatible with an identical value. This could lead to very cumbersome encodings being necessary, especially for selectional restrictions. One natural way round this was to allow values to be partially specified using *sorts* (e.g. Calder, 1989), which allowed hierarchies to be directly encoded. Full sorted or typed feature structure languages take this a step further, so that sorts (also known as types) are associated with every node in an FS, not just the terminal ones. Constraints can be specified on sorts: these must be satisfied by every FS which has that sort, or any of its subsorts. This, in effect, makes inheritance a built-in part of the feature structure language. HPSG (Pollard and Sag, 1987, 1994) uses this notion of sorts, but other linguistic frameworks can also be implemented within a sorted (typed) feature structure language.

Sort hierarchies with constraints differ fundamentally from templates in that the constraints on FSs are assumed to apply uniformly, however the FS arises, whereas templates only apply during a preliminary processing phase associated with the evaluation of FS descriptions. If we consider a description of a template as inheriting from two template FSs, the result will be subsumed by their unification, just as a subtype of two sorts will have a constraint which is subsumed by the unification of the constraints on its parents. But while templates are restricted to the descriptive metalanguage, types are a part of the object language. Thus, if we define two templates, Intrans-verb and Monadic as shown in Figure 1 and define Monadic-stative-intrans-verb to inherit from both of them, this does not affect the result of unification of two FSs which happen to be equal to the template descriptions. In contrast, consider the type system also shown in Figure 1, which implies that whenever two FSs of types **intrans-verb** and **monadic** are unified, the result must be of type **monadic-stative-intrans-verb**, and therefore, to be well-formed, must be subsumed by its constraint, which in this case is more specific than the unification of its parent constraints.

Confusingly, there are several versions of typed feature structure languages (including Carpenter, 1992; Gerdemann and King, 1994; Krieger and Schäfer, 1994; Emele and Zajac, 1990; Dörre and Eisle, 1991; Smolka, 1989; Copestake *et al*, 1993; Alshawi *et al*, 1991). Not all of these languages would allow the constraint specification shown in Figure 1. The function of sorts with respect to consistency checking is sometimes emphasised rather more than their role in providing inheritance. Recursive constraint specifications give systems enough power to implement parsers and generators directly, though not very tractably (see Carpenter, 1992; Emele and Zajac, 1990). Several of the systems mentioned above, however, compile out phrasal constraints to give conventional grammar rules, in order to allow standard parsing techniques to be used.

The systems mentioned above are all monotonic. If they allow for defaults at all, they do so as part of the description language: i.e. they allow some version of default templates as well as sorts.

3 Default inheritance

Many authors have argued for the utility of defaults in inheritance-based accounts of lexical organisation (e.g. Boguraev and Pustejovsky, 1990; Briscoe *et al*, 1990; Lascarides and Copestake, 1995a; Vossen and Copestake, 1993; Daelemans 1987; Evans and Gazdar 1989a,b; Flickinger *et al*, 1985; Flickinger, 1987; Flickinger and Nerbonne, 1992; Krieger and Nerbonne, 1993; Shieber, 1986a and others). One frequent motivation is to capture phenomena of blocking, or the overriding of regularities by subregularities (e.g. Calder, 1989; Copestake and Briscoe, 1992; Briscoe *et al*, 1995). Approaches to formalising and implementing inheritance-based lexicons include utilising object-oriented programming techniques (Daelemans, 1987; Flickinger *et al*, 1985), knowledge representation languages (Pustejovsky and Boguraev, 1993; Flickinger, 1987), overwriting templates (Karttunen, 1986; Shieber, 1986a) and various definitions of default unification (see below).

In order to provide an account of defaults which is closely integrated with the usual monotonic operations on FSs, it is natural to assume a notion of default unification. Most definitions (e.g. Bouma 1990, 1992; Copestake, 1993; Carpenter, 1993; Russell *et al*, 1991, 1993) have followed Kaplan (1987) in assuming that this is an asymmetric binary operation, with one default FS being regarded as adding consistent information to a non-default FS. The result is taken to be either a single FS or a disjunction of FSs. Although this asymmetry is useful for some applications, it is not ideal for lexical inheritance, because inheritance order must be stipulated, since default unification is not associative under any of these definitions. Apart from this, individual specifications of defaults which pertain to a class cannot be evaluated incrementally with an asymmetric operation because the information contributed by earlier defaults becomes non-default. Furthermore, non-associativity makes default unification unsuitable for use in situations where the application of defaults cannot be circumscribed.

Young and Rounds (1993), Lascarides *et al* (in press) and Lascarides and Copestake (1995b) attempt to address this problems by defining a symmetric, order independent version of default unification on typed feature structures. In these accounts, defeasible (i.e. default) information is marked explicitly in FSs rather than treating default unification as an asymmetric operation on a defeasible FS and an indefeasible FS. The defeasible information, if it survives unification, persists as defeasible information, via the explicit marking. Lascarides *et al* refer to this approach as 'persistent' default unification.

Lascarides and Copestake (1995b) argue that ideally, default unification should have the following characteristics:

1. Non-default information can be distinguished from default information and is always preserved.

2. Default unification behaves like monotonic unification in the cases where monotonic unification would succeed.

3. Default unification returns a single result, deterministically.

4. Defaults can be given a precedence ordering such that more specific information overrides less specific information. (Young and Rounds (1993) definition does not meet this criterion and Lascarides *et al* (in press) does not meet it completely.)

5. Default unification can be described using a binary, order independent (i.e., commutative and associative) operation.

6. Default unification never fails unless there is conflict in non-default information.

A trivial example of the use of the version of default unification described by Lascarides and Copestake (1995b) is given in Figure 2. This is a schematic representation of the morphology of English verbs where there is an attribute for the past tense form (PST) and another for the past particle (PSP). Normally these are the same, and this can be represented by default reentrancy (coindexation) as shown in the constraint for **regular-verb**. The notation uses a slash to delimit the default part of the FS from the non-default. The PST and PSP forms differ in some classes of irregular verbs (e.g. *speak, spoke, spoken*), here given as **psp-diff-vb**, and thus the reentrancy has to be overridden, but the specification of the vowel change and suffixation for this class is itself stated as defeasible, because of verbs such as *weave* which are also found with the regular suffixation pattern (*wove/weaved, woven/weaved*).

The constraints on types are thus a combination of non-default and default information, and the specificity ordering on the defaults is given by the type hierarchy. In this approach, therefore, both non-default and default inheritance is fully integrated into the object language. This therefore differs from

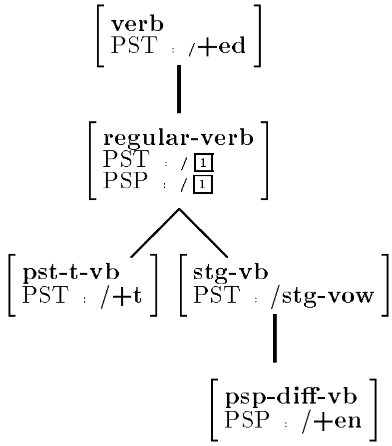

Figure 2: Default Inheritance Hierarchy

systems such as DATR, where the default description language is separate from the feature structure language. I have given the morphology example for expository purposes, but for more motivation for the use of persistent defaults, see Lascarides *et al* (in press) and also Lascarides and Copestake (1995a).

References

Alshawi, H., D.J. Arnold, R. Backofen, D.M. Carter, J. Lindop, K. Netter, S.G. Pulman, J. Tsujii, H. Uszkoreit (1991) *Eurotra ET6/1: rule formalism and virtual machine design study (final report),* CEC, Luxembourg.

Boguraev, B. and J. Pustejovsky (1990) 'Lexical ambiguity and the role of knowledge representation in lexicon design', *Proceedings of the 13th Int. Conf. on Comp. Ling. (COLING-90),* Helsinki, pp. 36–42.

Bouma, G. (1990) 'Defaults in unification grammar', *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (ACL-90),* Pittsburg, Pennsylvania, pp. 165–173.

Bouma, G. (1992) 'Feature structures and nonmonotonicity', *Computational Linguistics, vol.18.2,* 183–204.

Briscoe, E.J., A. Copestake and B.K. Boguraev (1990) 'Enjoy the paper: Lexical semantics via lexicology', *Proceedings of the 13th Int. Conf. on Comp. Ling. (COLING-90),* Helsinki, pp. 42–47.

Briscoe, E.J., A. Copestake and V. de Paiva (eds) (1993) *Inheritance, defaults and the lexicon,* Cambridge University Press, Cambridge, England.

Briscoe, E.J., A. Copestake and A. Lascarides (1995) 'Blocking' in P. St. Dizier, P. and E. Viegas (eds.), *Computational lexical semantics,* Cambridge University Press, Cambridge, England, pp. 273–302.

Calder, J. (1989) 'Paradigmatic morphology', *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics (EACL-89),* Manchester, England, pp. 58–65.

Carpenter, R. (1992) *The logic of typed feature structures,* (Tracts in Theoretical Computer Science), Cambridge University Press, Cambridge, England.

Carpenter, R. (1993) 'Skeptical and credulous default unification with application to templates and inheritance' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, defaults and the lexicon,* Cambridge University Press, Cambridge, England, pp. 13–37.

Copestake, A. (1993) 'Defaults in lexical representation' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, defaults and the lexicon,* Cambridge University Press, Cambridge, England, pp. 223–245.

Copestake, A. (1993) *The representation of lexical semantic information,* Doctoral dissertation, University of Sussex, available as Cognitive Science Research Paper CSRP 280.

Copestake, A. and E.J. Briscoe (1995) 'Semi-productive polysemy and sense extension', *Journal of Semantics, vol.12:1,* 15–67.

Copestake, A., A. Sanfilippo, E.J. Briscoe and V. de Paiva (1993) 'The ACQUILEX LKB: an introduction' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, defaults and the lexicon,* Cambridge University Press, Cambridge, England, pp. 148–163.

Daelemans, W. (1987) 'A tool for the automatic creation, extension and updating of lexical knowledge bases', *Proceedings of the 3rd Conference of the European Chapter of the Association for Computational Linguistics (EACL-87),* Copenhagen, pp. 70–74.

Daelemans, W., K. de Smedt and G. Gazdar (1992) 'Inheritance in natural language processing', *Computational Linguistics, vol.18.2,* 205–218.

Dörre, J. and A. Eisle (1991) *A comprehensive unification-based grammar formalism,* DYANA Technical Report, University of Edinburgh.

Emele, M. and R. Zajac (1990) 'Typed unification grammars', *Proceedings of the 13th Int. Conf. on Comp. Ling. (COLING-90),* Helsinki, pp. 293–298.

Evans, R. and G. Gazdar (1989a) 'Inference in DATR', *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics (EACL-1989),* Manchester, England, pp. 66–71.

Evans, R. and G. Gazdar (1989b) 'The semantics of DATR' in A. G. Cohn (eds.), *Proceedings of the Seventh Conference of the Society for the Study of Artificial Intelligence and Simulation of Behavior (AISB),* Pitman/Morgan Kaufmann, London, pp. 79–87.

Flickinger, D. (1987) *Lexical rules in the hierarchical lexicon,* PhD thesis, Stanford University.

Flickinger, D., C. Pollard and T. Wasow (1985) 'Structure sharing in lexical representation', *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics (ACL-85),* University of Chicago, pp. 262–268.

Flickinger, D. and J. Nerbonne (1992) 'Inheritance and complementation: A case study of *easy* adjectives and related nouns', *Computational Linguistics, vol.18.3,* 269–310.

Gerdemann, D. and P. King (1994) 'The correct and efficient implementation of appropriateness specifications for typed feature structures', *Proceedings of the 15th Int. Conf. on Comp. Ling. (COLING-94),* Kyoto, Japan.

Kaplan, R. (1987) 'Three seductions of computational psycholinguistics' in P. Whitelock, H. Somers, P. Bennett, R. Johnson, and M. McGee Wood (eds.), *Linguistic theory and computer applications,* Academic Press, London, pp. 149–188.

Karttunen, L. (1986) 'D-PATR: a development environment for unification-based grammars', *Proceedings of the 11th Int. Conf. on Comp. Ling. (COLING-86),* Bonn, Germany, pp. 74–80.

Kreiger, H. and J. Nerbonne (1993) 'Feature-based inheritance networks for computational lexicons' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, defaults and the lexicon,* Cambridge University Press, Cambridge, England, pp. 90–136.

Krieger, H-U, and U. Schäfer (1994) *TDL — A type description language for HPSG,* DFKI, Saarbrücken, Germany.

Lascarides, A. and A. Copestake (1995a) 'The pragmatics of word meaning', *Proceedings of the SALT V,* Cornell University.

Lascarides, A. and A. Copestake (1995b) *A logic for order independent typed default unification,* ACQUILEX research report.

Lascarides, A., E.J. Briscoe, N. Asher and A. Copestake (1995) 'Order independent and persistent typed default unification', *Linguistics and Philosophy,*

Pollard, C. and I. Sag (1987) *An information-based approach to syntax and semantics: Volume 1 fundamentals,* CSLI Lecture Notes 13, Stanford CA.

Pollard, C. and I. Sag (1994) *Head-driven Phrase Structure Grammar,* CSLI lecture notes series, Chicago University Press, Chicago.

Pustejovsky, J. and B. Boguraev (1993) 'Lexical knowledge representation and natural language processing', *Artificial Intelligence, vol.63,* 193–223.

Russell, G., J. Carroll and S. Warwick (1991) 'Multiple default inheritance in a unification-based lexicon', *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL-91),* Berkeley, California, pp. 215–221.

Russell, G., A. Ballim, J. Carroll and S. Warwick-Armstrong (1993) 'A practical approach to multiple default inheritance for unification-based lexicons' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, defaults and the lexicon,* Cambridge University Press, Cambridge, England, pp. 137–147.

Sanfilippo, A. (1993) 'LKB encoding of lexical knowledge' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, defaults and the lexicon,* Cambridge University Press, Cambridge, England, pp. 190–222.

Shieber, S. M. (1986) *An introduction to unification-based approaches to grammar,* CSLI Lecture Notes 4, Stanford CA.

Smolka, G. (1989) *Feature constraint logic for unification grammars,* IWBS Report 93, IWBS-IBM, Stuttgart, Germany.

Vossen, P. and A. Copestake (1993) 'Untangling definition structure into knowledge representation' in E. J. Briscoe, A. Copestake and V. de Paiva (eds.), *Inheritance, Defaults and the Lexicon,* Cambridge University Press, Cambridge, England, pp. 246–274.

Young, M. and W. Rounds (1993) 'A logical semantics for nonmonotonic sorts', *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL-93),* Columbus, Ohio, pp. 209–215.

ACQUILEX Technical reports can be obtained by emailing a request to `cide@cup.cam.ac.uk` — many ACQUILEX papers are also available by ftp from `ftp://ftp.cl.cam.ac.uk/papers/acquilex/`