The ACQUILEX LKB: representation issues in semi-automatic acquisition of large lexicons

Ann Copestake

University of Cambridge Computer Laboratory
New Museums Site, Pembroke Street, Cambridge, CB2 3QG, UK

January 1992

(in Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP-92), Trento, Italy)

ESPRIT BRA-3030 ACQUILEX Working Paper No. 036

The ACQUILEX LKB: representation issues in semi-automatic acquisition of large lexicons

Ann Copestake
University of Cambridge Computer Laboratory
New Museums Site, Pembroke Street, Cambridge, CB2 3QG, UK
Ann.Copestake@cl.cam.ac.uk

January 1992

Abstract

We describe the lexical knowledge base system (LKB) which has been designed and implemented as part of the ACQUILEX project[1] to allow the representation of multilingual syntactic and semantic information extracted from machine readable dictionaries (MRDs), in such a way that it is usable by natural language processing (NLP) systems. The LKB's lexical representation language (LRL) augments typed graph-based unification with default inheritance, formalised in terms of default unification of feature structures. We evaluate how well the LRL meets the practical requirements arising from the semi-automatic construction of a large scale, multilingual lexicon. The system as described is fully implemented and is being used to represent substantial amounts of information automatically extracted from MRDs.

1 Introduction

The ACQUILEX LKB is designed to support representation of multilingual lexical information extracted from machine readable dictionaries (MRDs) in such a way that it can be utilised by NLP systems. In contrast to lexical database systems (LDBs) or thesaurus-like representations (e.g. Alshawi *et al.*, 1989; Calzolari, 1988) which represent extracted data in such a way as to support browsing and querying, our goal is to build a knowledge base which can be used as a highly structured reusable lexicon, albeit one much richer in lexical semantic information than those commonly used in NLP. Thus, although we are using information which has been derived from MRDs (possibly after considerable processing involving some human intervention), our aim is not to represent the dictionary entries themselves. Our methodology is to store the dictionary entries and raw extracted data in our LDB (Carroll, 1990) and to use this information to build LKB entries which could be directly utilised by an NLP system. Briscoe (1991) discusses the LDB/LKB distinction in more detail and describes the ACQUILEX project as a whole.

Practical NLP systems need large lexicons. Even in cases such as database front ends, where the domain of the application is highly restricted, a practical natural language interface must be able to cope with an extensive vocabulary, in order to respond helpfully to a user who lacks domain knowledge, for example. For applications such as text-to-speech synthesis, interfaces to large-scale knowledge based systems, summarising and so on, large lexicons are clearly needed; for machine translation the requirement is for a large scale, multilingual lexical resource. Acquisition of such information is a serious bottleneck in building NLP systems, and MRD sources currently seem the most promising source for semi-automatically acquiring the syntactic and semantic information needed.

Previous work on extracting and representing syntactic information includes the work done on the Alvey Tools lexicon project (Carroll and Grover 1989) in which a large scale lexicon was produced semi-automatically from LDOCE (Longman Dictionary of Contemporary English, Procter,

[1] 'The Acquisition of lexical knowledge for Natural Language Processing systems' (Esprit BRA-3030)

1978) using a feature and unification based representation. There has been considerable discussion and some implementation of LKBs for the representation of semantic information extracted from MRDs (e.g. Boguraev and Levin, 1990; Wilks *et al*, 1989). However the knowledge representation languages assumed are rarely described formally; typically a semantic network or a frame representation has been suggested, but the interpretation and functionality of the links has been left vague. Several networks based on taxonomies have been built, and these are useful for tasks such as sense-disambiguation, but are not directly utilisable as NLP lexicons. For a reusable lexicon, a declarative, formally specified, representation language is essential.

In the ACQUILEX project we are concerned with the extraction and representation of both syntactic and lexical semantic information. A common representation language is needed, to allow the interaction of lexical semantic and syntactic properties to be described. There is currently a considerable amount of work in lexical semantics where unification based formalisms are used to represent this interaction (e.g. Briscoe *et al.*'s (1990) account of logical metonymy (Pustejovsky, 1989, 1991), Sanfilippo's (1990) representation of thematic and aspectual information). However we also wish to structure the lexicon, in order to link lexical entries. This is essential, since we are ultimately considering lexicons with maybe 100,000 entries for each language. Although the aim of the ACQUILEX project is to determine the feasibility of using MRD sources, rather than attempting to build a lexicon of such size, we nevertheless need an LKB which can cope with tens of thousands of entries.

There are currently several approaches to developing representation languages which allow the lexicon to be structured, in particular by inheritance. These include object-oriented approaches (Daelemans, 1990), and DATR (Evans and Gazdar, 1990). We chose to use a graph unification based representation language for the LKB, because this offered the flexibility to represent both syntactic and semantic information in a way which could be easily integrated with much current work on unification grammar, parsing and generation. In contrast to DATR for example, the LKB's representation language (LRL) is not specific to lexical representation. This made it much easier to incorporate a parser in the LKB (for testing lexical entries) and to experiment with notions such as lexical rules and interlingual links between lexical entries. Although this means that the LRL is in a sense too general for its main application, the typing system provides a way of constraining the representations, and the implementation can then be made more efficient by taking advantage of such constraints.

Our typed feature structure mechanism is based on Carpenter's work on the HPSG formalism (Carpenter 1990) although there are some significant differences. We augment the formalism with the more flexible *psort* inheritance mechanism, which allows for default inheritance. Much of the motivation behind this comes from consideration of the sense-disambiguated taxonomies semi-automatically derived from MRDs, which we are using to structure the LKB (see Copestake 1990). The notion of types, and features appropriate for a given type, gives some of the properties of frame representation languages, and allows us to provide a well-defined, declarative representation, which integrates relatively straightforwardly with much current work on natural language processing and lexical semantics.

Thus the operations that the LKB supports are (default) inheritance, (default) unification, and lexical rule application. It does not support any more general forms of inference and is thus designed specifically to support processes which concern lexical rather than general reasoning. In the rest of this paper we first informally introduce the way in which lexical entries are represented in the LKB. We then describe the LRL, and discuss how the design of the default inheritance system was influenced by the application. (A fuller and more formal account of the LRL appears in papers in Briscoe *et al.*, forthcoming.) We conclude with an overview of the actual implementation and a discussion of the utility of typed feature structures and the psort mechanism in practise.

2 Lexical entries in the LKB

Consider Figure 1, which is a screen dump of the LKB system showing part of a file containing a semi-automatically generated lexical entry for the Dutch noun *kippevlees* (chicken meat) (top right of figure), the feature structure (FS) representation of that description (bottom right) and

Figure 1: A lexical entry

the fully typed feature structure into which it is expanded in the LKB (left of figure). See Vossen (1991) for the details of the generation of this entry. Features are shown uppercased, types are in lowercase bold, reentrancy is indicated by numbers in angle brackets. The identifier for the lexical entry, `kippevlees_V_0_1`, indicates that it corresponds to the sense kippevlees 1 in the Van Dale dictionary. The unexpanded lexical entry is relatively compact, but a large amount of information is inherited via the type and psort systems. The expanded lexical entry is not shown completely; the entry's syntactic type is **noun-cat**, and the box round this indicates that its internal structure is not displayed. The same applies to the sense-id information (which enables the corresponding LDB entry to be accessed) and the argument structure. Figure 2 (left) shows the type **lex-uncount-noun**, which determines the basic skeleton of the entry. Feature structures (called *constraints*) are associated with types and inherited by all FSs of a particular type. Thus the form of this lexical entry is due to the constraint on **lex-uncount-noun** shown in the figure.

Default inheritance from the lexical semantic structure for the lexical entry for **vlees_V_0_1** augments the type information for the entry for *kippevlees*. We encode a relatively rich lexical semantic structure for nouns (referred to as the 'relativised qualia structure', RQS) based on the notion of *qualia structure*, described by Pustejovsky (1989, 1991). Noun lexical entries are parsed to yield a *genus term*, *vlees* in this case, and *differentia*. The genus term is normally interpreted in LKB terms as specifying the lexical entry from which information is inherited by default; as explained in Section 4 this also partially defines the lexical semantic type (RQS type), which is **c_nat_subst** in this example (for comestible, natural, substance). A fragment of the RQS type hierarchy is also shown in Figure 2 [2]. The differentia can be partially interpreted relative to the

[2] Unlike Carpenter(1990) we adopt a notation with the most general type at the top of any diagram, because this

Figure 2: Types

RQS type; in this example `< rqs : origin > = "kip"` is an indication that *kippevlees* comes from *kip*; eventually this will allow their lexical entries to be linked automatically by the appropriate lexical rule (Copestake and Briscoe, 1991; Copestake *et al.*, 1992). The feature ORIGIN is introduced at type **natural** (the FS definition of **natural** is shown in Figure 2, top right, before expansion by inheritance from **nomrqs**). Since **natural** is a parent of **c_nat_subst**, ORIGIN is an *appropriate feature* for **c_nat_subst**.

The feature TELIC is used to provide a slot for the semantics of the verb sense which is associated with the purpose of an entity (*eating* in this case). The way in which such a representation may be used in the treatment of logical metonymy was described in Briscoe et al (1990). Other features (such as PHYSICAL-STATE) are used to encode information which is useful for applications such as sense-disambiguation. This attempt to represent detailed lexical semantic information illustrates a general principle of the ACQUILEX project; such lexical entries are usable by a wide range of NLP systems because they are relatively rich and detailed; applications which do not make use of detailed lexical semantic information can simply discard the information. Clearly the converse is not true, and a more impoverished representation would be less generally useful. We thus aim for representations which are as rich as possible in information which we can extract automatically, and represent formally, but which are also well motivated linguistically and/or useful for practical NLP applications. This also applies to our use of thematic roles in the semantics; see the examples

seems more natural to the main users of the system.

of LKB entries for verbs given in Sanfilippo and Poznanski (1992, this volume).

3 The type system

In the definition of a type hierarchy we follow Carpenter(1990) very closely. The type hierarchy defines a partial order (notated $\sqsubseteq$, "is more specific than") on the types and specifies which types are *consistent*. Only FSs with mutually consistent types can be unified — two types which are unordered in the hierarchy are assumed to be inconsistent unless the user explicitly specifies a common subtype. Every *consistent* set of types $S \subseteq \mathsf{TYPE}$ must have a unique greatest lower bound or meet (notation $\sqcap S$). [3] This condition allows FSs to be typed deterministically — if two FSs of types **a** and **b** are unified the type of the result will be $\mathbf{a} \sqcap \mathbf{b}$, which must be unique if it exists. If $\mathbf{a} \sqcap \mathbf{b}$ does not exist unification fails. In the fragment of a type hierarchy shown in Figure 2 **c_natural** and **natural_substance** are consistent; **c_natural** $\sqcap$ **natural_substance** = **c_nat_subst**. Because the type hierarchy is a partial order it has properties of reflexivity, transitivity and anti-symmetry (from which it follows that the type hierarchy cannot contain cycles).

We define a typed feature structure as a tuple $F = \langle Q, q_0, \delta, \theta \rangle$, where the only difference from the untyped case is that every node of a typed FS has a type, $\theta(q)$. The type of a FS is the type of its initial node, $\theta(q_0)$. The definition of subsumption of typed FSs is very similar to that for untyped FSs, with the additional proviso that the ordering must be consistent with the ordering on their types. We thus overload the symbol $\sqsubseteq$ ("is-more-specific-than", "is-subsumed-by") to express subsumption of FSs as well as the ordering on the type hierarchy. Thus if F_1 and F_2 are FSs of types t_1 and t_2 respectively, then $F_1 \sqsubseteq F_2$ only if $t_1 \sqsubseteq t_2$.

3.1 Constraints

Our system differs somewhat from that described by Carpenter in that we adopt a different notion of well-formedness of typed feature structures. In our system every type must have exactly one associated FS which acts as a constraint on all FSs of that type; by subsuming all well-formed FSs of that type. The constraint also defines which features are *appropriate* for a particular type; a well-formed FS may only contain appropriate features. Constraints are inherited by all subtypes of a type, but a subtype may introduce new features (which will be inherited as appropriate features by all its subtypes). A constraint on a type is a well-formed FS of that type; all constraints must therefore be mutually consistent.

Features may only be introduced at one point in the type hierarchy (cf Carpenter's minimal introduction). Because of the condition that any consistent set of types must have a unique greatest lower bound, it is also the case that sets of features will become valid at unique greatest points in the type hierarchy. This allows undertyped feature structures to be introduced into the system by the user which are then given the most general possible type. The importance of this form of type inference for our application is discussed in Section 5.2, below.

Constraints are given by the function

$$C\colon \langle \mathsf{TYPE}, \sqsubseteq \rangle \rightarrow \mathcal{F}$$

where $\mathcal{F}$ is the set of FS. $C(t)$ denotes the constraint FS associated with type t. We define the notion of appropriate features as follows:

Definition 1 *If $C(t) = \langle Q, q_0, \delta, \theta \rangle$ we define $Appfeat(t) = Feat(q_0)$ where we define $Feat(q)$ to be the set of features labelling transitions from the node q such that $f \in Feat(q)$ if $\delta(f, q)$ is defined.*

The conditions on the constraint function are as follows:

[3] In order to check the type hierarchy for uniqueness of greatest lower bounds we carry out a pairwise comparison on types with multiple parents to see if they have a unique lowest greater bound. Since the number of types with multiple parents is typically much less than the total number of types, this is considerably more efficient than carrying out pairwise comparisons on all the types in the hierarchy.

Monotonicity Given types t_1 and t_2, if $t_1 \sqsubseteq t_2$ then $C(t_1) \sqsubseteq C(t_2)$

Type For a given type t, if $C(t)$ is the FS $\langle Q, q_0, \delta, \theta \rangle$ then $\theta(q_0) = t$.

Consistency of constraints For all $q \in Q$, we have that $F' = \langle Q', q, \delta, \theta \rangle \sqsubseteq C(\theta(q))$ and $Feat(q) = Appfeat(\theta(q))$.

We therefore disallow any occurrence of t in a substructure of $C(t)$, thus if $C(t) = \langle Q, q_0, \delta, \theta \rangle$ then for all $q \in Q$, $q \neq q_0$ implies that $\theta(q) \neq t$. Since we disallow cycles in FSs such a constraint could only be satisfied by an infinite FS, which is also disallowed.

Maximal introduction of features For every feature $f \in \mathsf{FEAT}$ there is a unique type $t = Maxtype(f)$ such that $f \in Appfeat(t)$ and there is no type s such that $t \sqsubset s$ and $f \in Appfeat(s)$. The maximal appropriate value of a feature $Maxappval(f)$ is the type t such that if $C(Maxtype(f)) = \langle Q, q_0, \delta, \theta \rangle$ then $t = \theta(\delta(f, q_0))$

Definition 2 *We say that a given FS $F = \langle Q, q_0, \delta, \theta \rangle$ is a well-formed FS iff for all $q \in Q$, we have that $F' = \langle Q', q, \delta, \theta \rangle \sqsubseteq C(\theta(q))$ and $Feat(q) = Appfeat(\theta(q))$.*

Carpenter separates the notion of typing and constraints. This allows a more powerful constraint language, but complicates the system. Since the users of the LKB were initially not familiar with feature structure representations it was important to keep the system as simple as possible, and in practice we have not yet found the additional power of Carpenter's constraint language necessary.

Some relatively minor extensions to the formalism allow the implementation of some cooccurrence restrictions and the disjunction of atomic types. It is necessary to allow types with string values, representing orthography for example, to be introduced as needed rather than predefined; we therefore define an atomic type **string** which is allowed to have any string as a subtype without these being explicitly specified. All subtypes of **string** are taken to be disjoint.

4 Default inheritance and taxonomies

We extend the typed FS system with default inheritance. FSs may be specified as inheriting by default from one or more other (well-formed) FSs which we refer to in this context as *psorts*. Psorts may correspond to (parts of) lexical entries or be specially defined. Since psorts may themselves inherit information, default inheritance (notated by $<$, "inherits from") in effect operates over a hierarchy of psorts. We prohibit cycles in the inheritance ordering. Inheritance order must correspond to the type hierarchy order.

$p_1 < p_2 \Rightarrow Typeof(p_1) \sqsubseteq Typeof(p_2)$
where p_1 and p_2 are psorts

The typing system thus restricts default inheritance essentially to the filling in of values for features which are defined by the type system.

Default inheritance is implemented by a version of default unification, for a detailed discussion of which see Carpenter (1991, forthcoming). In default unification, unlike ordinary unification, inconsistent information is ignored rather than causing failure; however the definition is complicated by the need to consider the interactions between reentrant FSs. The way we deal with this is discussed in detail in Copestake(1991, forthcoming), but since the problematic cases seem to arise relatively rarely in our particular application, we will not discuss the full definition here. We use $\sqcap_<$ to signify default unification, where $A \sqcap_< B$ means that A is the non-default and B the default FS. When no reentrancy interactions are involved the definition is:

$$A \sqcap_< B = A \sqcap \bigsqcap \{\psi \in \Psi \mid A \sqcap \psi \neq \perp\}$$

where Ψ is the set of all component FSs of B.

The ordering on the psort hierarchy gives us an ordering on defaults. So for example, assume that the following is the lexical entry for BOOK_L_1_1:

$$\begin{bmatrix} \textbf{lex-noun-sign} & \\ \text{RQS} = & \begin{bmatrix} \textbf{artifact\_physical} & \\ \text{TELIC} = & \begin{bmatrix} \textbf{verb-sem} \\ \text{PRED} = \textbf{read\_L\_1\_1} \end{bmatrix} \\ \text{PHYSICAL-STATE} = \textbf{solid\_a} & \end{bmatrix} \end{bmatrix}$$

The following path specifications make the lexical entries defined inherit from BOOK_L_1_1:

```
autobiography <rqs> < book_L_1_1 <rqs>
dictionary    <rqs> < book_L_1_1 <rqs>
              <rqs : telic : pred >
                     = refer_to_L_0_2
lexicon       <rqs> < dictionary <rqs>
```

AUTOBIOGRAPHY would thus have the same values as BOOK_L_1_1 for both telic and physical-state. DICTIONARY will inherit the value **solid_a** for the feature PHYSICAL-STATE but the value of TELIC overrides that inherited from BOOK_L_1_ 1. LEXICON inherits its value for the telic role from DICTIONARY rather than from BOOK_L_1_1:

$$\begin{bmatrix} \textbf{lex-noun-sign} & \\ \text{RQS} = & \begin{bmatrix} \textbf{artifact\_physical} & \\ \text{TELIC} = & \begin{bmatrix} \textbf{verb-sem} \\ \text{PRED} = \textbf{refer\_to\_L\_0\_2} \end{bmatrix} \\ \text{PHYSICAL-STATE} = \textbf{solid\_a} & \end{bmatrix} \end{bmatrix}$$

Multiple default inheritance is allowed but is restricted to the case where the information from the parent psorts does not conflict. This is enforced by unifying all (fully-expanded) immediate parent psorts before default unifying the result with the daughter psort. The type restriction on default inheritance means that all the psorts must have compatible types and the type of the daughter must be the meet of those types. We define inheritance to operate top-down; that is a psort will be fully expanded with inherited information before it is used for default inheritance. We adopted this approach as we are primarily interested in default inheritance between fully formed lexical entries; since we disallow conflicts arising from multiple inheritance, distinctions between top-down and bottom-up inheritance only arise with the problematic cases of default unification alluded to above.

We also allow non-default inheritance from psorts, implemented by ordinary unification. This is a relatively recent addition to the LKB, prompted partly by issues in the representation of the multilingual translation links. It also seemed to be desirable in the representation of qualia structure, in order to allow the telic role of a noun to be specified directly in terms of a verb sense, without allowing other information in that lexical entry to conflict. Thus the entry for dictionary above would actually specify:

```
<rqs : telic > == refer_to_L_0_2 < sem >
```

where == indicates non-default inheritance.

Although introducing psorts as well as types may seem unnecessarily complex there seem to be compelling reasons for doing so for this application, where we wish to use taxonomic information extracted from MRDs to structure the lexicon. The type hierarchy is not a suitable way for representing taxonomic inheritance for several reasons. Perhaps the most important is that taxonomically inherited information is defeasible, but typing and defaults are incompatible notions. Types are needed to enforce an organisation on the lexicon — if this can be overridden it is useless. Furthermore the type system is taken to be complete, and various conditions are imposed on it, such as the greatest lower bound condition, which ensure that deterministic classification is possible. Taxonomies extracted from dictionaries will not be complete in this sense, and will not meet these conditions. Intuitively we would expect to be able to classify lexical entries into categories such as **human**, **artifact** and so on, and to be able to state that all **creature**s are either **human**s or **animal**s, since in effect this is how we are defining those types. But we would not expect to be able to use the finer-grained, automatically acquired information in this way; we will never extract all possible categories of *horse* for example.

In implementational terms, using the type hierarchy to provide the fine-grain of inheritance possible with taxonomic information would be very difficult. A type scheme should be relatively static; any alterations may affect a large amount of data and checking that the scheme as a whole is still consistent is a non-trivial process. Because the inheritance hierarchies are derived from taxonomies and thus are derived semi-automatically from MRDs, they will contain errors and it is important that these can be corrected easily. In practise, deciding whether to make use of the type mechanism or the psort mechanism has been relatively straightforward. If we wish to use a feature which is particular to some group of lexical entries we have to introduce a type, otherwise, especially if the information might be defeasible, we use a psort.

Several of the decisions involved in designing the default inheritance system were thus influenced by the application. The condition that the default inheritance ordering reflects the type ordering was partly motivated by the desire to be able to provide an rqs type for lexical entries on the basis of taxonomic data alone. However it also seems intuitively reasonable as a way of restricting default inheritance; without some such restriction it is difficult to make any substantive claims when default inheritance is used to model some linguistic phenomenon.

5 Using the LKB

5.1 Interface and implementation

The LKB as described here is fully implemented in Procyon Common Lisp running on Apple Macintoshes. It is in use by all the academic groups involved in the ACQUILEX project. In total there are currently about 20 users on five sites in different countries. Interaction with the LKB is entirely menu-driven. Besides the obvious functions to load and view types, lexical entries, psorts, lexical rules and so on, there are various other facilities which are necessary for the application. A very simple (and inefficient) parser is included, to aid development of types and lexical entries. There are tools for supporting multilingual linked lexicons, described in Copestake *et al.* (1992). The LKB is integrated with our LDB system so that information extracted from dictionary entries stored in the LDB can be used to build LKB lexicons.

The type system which has been developed for use on the ACQUILEX project is fairly large (about 450 types and 80 features). Currently nearly 15,000 lexical entries containing syntactic and semantic information have been stored in the LKB. The bulk of these entries are currently made up of nouns for which the main semantic information is inherited down semi-automatically derived taxonomies. Sanfilippo and Poznanski (1992) describe the semi-automatic derivation of entries for English psychological predicates by augmenting LDOCE with thesaurus information derived from the Longman Lexicon. Work has begun on deriving multi-lingual linked lexicons.

Given the complexity of the FSs for lexical entries, and the size of the lexicons to be supported by the LKB, it is clearly not possible to store lexicons in main memory. Lexical entries are thus stored on disk, to be expanded as required. Entries may be indexed by type of FS at the end of user-defined paths, and also by the psort(s) from which they are defined to inherit, although producing such indices for large lexicons is time consuming. Checking lexical entries (for well-formedness, default inheritance conflicts and presence of cycles) can be carried out at the same time as indexing or acquisition.

Efficiency gains arising directly from the use of types were not a major factor in our decision to use a typed system. Although parsing with typed FSs is more efficient than with untyped ones, since unification will fail when type conflict occurs, this is not particularly important in the LKB, since most unifications will be performed while expanding lexical entries, when the vast majority of unifications would be expected to succeed. Since there is some overhead in typing the FSs, the use of types probably decreases efficiency slightly, although the unifications involved will be comparable to those needed if the same information were conveyed by templates. Since the LKB has to cope with large lexicons, with thousands of complex lexical entries, space efficiency rather than speed is the major consideration. The most important factor in space efficiency is the use of inheritance, both in the type system and the psort system, which allows unexpanded lexical entries to be very compact.

5.2 Typing and automatic acquisition of large lexicons

Our notion of typing of FSs can be regarded as a way of getting the functionality of templates in untyped FS formalisms, with the added advantages of type checking and type inference. As a method of lexical organisation, types have significant advantages over templates, especially for a large scale collaborative project. Once an agreed type system is adopted, the compatability of the data collected by each site is guaranteed. There may of course be problems of differing interpretation of types and features, but this applies to any representation; to ameliorate them we associate short documentation information with each type, accessible via the menu interface from any point where the type is displayed. In an untyped feature system, typographical errors and so on may go undetected, and debugging a large template system can be extremely difficult; a type system makes error detection much simpler. Since a given FS has a type permanently associated with it, is also much more obvious how information has come to be inherited than if templates are used.

Essentially the same advantages of safety and clarity apply to strict typing of FSs as to strict typing in programming languages. Of course a reduction in flexibility of representation has to be accepted, once a particular type system is adopted. It is possible to achieve a very considerable degree of modularisation; we have found that we could develop the noun RQS type system almost completely independently of the verb type system, once a small number of common types were agreed on, and that name clashes were the only problem found when reintegrating the two. After approximately eight months of use we are now on the third version of both the verb and the noun type systems; individual users have been experimenting with various representations which are then integrated into the general system as appropriate. Encoding the agreed representation in terms of a type system, rather than by means of templates, makes global alterations relatively easy because of the localisation of the information (for example, since a feature can only be introduced at one point in the hierarchy, it is easy to find all types which will be affected by a change in feature name) and the error checking. It is important that reprocessing of raw dictionary data is avoided when a type system is changed, particularly if user interaction is involved, but storing intermediate results in the LDB as a derived dictionary helps achieve this. Even within the project it has proved useful to have local type systems and lexicons, and to derive entries for these automatically from the general LKB. Currently this is achieved by ad-hoc methods; we intend to investigate the development of tools to make transfer of information easier and more declarative.

Ageno *et al.* (1992) describe one way in which the type system can be integrated with tools for semi-automatic analysis of dictionary definitions. Types are correlated with the templates used in a robust pattern matching parser, and user interaction can be controlled by the type system. The user is only allowed to introduce information appropriate for a particular type, and a menu-based interface can both inform the user of the possible values and preclude errors.

The utility of typing for error checking when representing automatically acquired data can be seen in the following simple example. The machine readable version of LDOCE associates semantic codes with senses. Examples of such codes are P for plant, H for human, M for male human, K for male human or animal, and so on. When automatically acquiring information about nouns from LDOCE, we specify a value for the feature SEX, where this is possible according to the semantic codes. Thus the automatically created lexical entry for *bull*[1] 1 contains the line:

```
< rqs : sex > = male
```

In the current type system the feature SEX is introduced at type **creature**. A few LDOCE entries have incorrect semantic codes; **Irish stew** for example has code K. Since **Irish stew** has RQS type **c_artifact**, which is not consistent with **creature**, SEX was detected as an inappropriate feature. Attempts at expansion of the automatically generated lexical entry caused an error message to be output, and the user had the opportunity to correct the mistake. If the LKB were not a typed system, errors such as this would not be detected automatically in this way.

In contrast, automatic classification of lexical entries by type, according to feature information, can be used to force specification of appropriate information. A lexical entry which has not been located in a taxonomy will be given the most general possible type for its RQS. However if a value for the feature SEX has been specified this forces an RQS type of **creature**. This would also force the value of ANIMATE to be **true**, for example.

5.3 The psort inheritance mechanism.

Manual association of information with psorts has proved to be a highly efficient method of acquiring information, since many psorts have hundreds of daughter entries. Creating 'artificial' psorts, which can be used where there is no simple lexicalisation of a concept, is also a powerful technique. Disjunctions such as *person or animal*, for example, can be represented as the generalisation of the two psorts involved. This and other cases of more complex taxonomic inheritance are discussed by Vossen and Copestake (1991, forthcoming).

We adopted the most conservative approach to multiple default inheritance (i.e. information inherited from multiple parents has to be consistent) because we knew we would have to cope with errors in extraction of information from MRDs, and with the lexicographers' original mistakes. We expected this to be overrestrictive, but in fact our consistency condition seems to be met fairly naturally by the data. Taxonomies extracted from MRDs are in general tree-structured (once sense-disambiguation has been performed); there do not tend to be many examples of genuine conjunction, for example. Multiple inheritance is mainly needed for cross-classification; artifacts for example may be defined principally in terms of their form or in terms of their function, but here different sets of features are typically specified, so the information is consistent. Furthermore it frequently turns out to be difficult to identify a second psort parent from the dictionary definition differentia. However type inference resulting from feature instantiation may still force a type to be assigned which represents the cross-classification.

Acknowledgements

Several people contributed in various ways to the design, implementation and development of the LKB, especially Valeria de Paiva, Antonio Sanfilippo, Ted Briscoe, John Carroll, John Bowler and Horacio Rodriguez. We are very grateful to Bob Carpenter for his detailed comments on our use of types and default unification. We are grateful to the publishers Longman, Biblograf, Van Dale and Garzanti for allowing groups involved in ACQUILEX to use their dictionaries.

References

A. Ageno *et al.*. SEISD: An Environment for Extraction of Semantic Information from On-Line Dictionaries. In *Proceedings of the 3rd Conference on Applied Natural Language Processing,* Trento, Italy, 1992.

H. Alshawi, B. Boguraev and D. Carter. Placing the dictionary on-line. In B. Boguraev and T. Briscoe (eds.), *Computational lexicography for natural language processing,* pages 41–63, Longman, London, 1989.

B. Boguraev and B. Levin. Models for lexical knowledge bases. In *Proceedings of the 6th Annual Conference of the UW Center for the New OED,* pages 65–78, Waterloo, 1990.

T. Briscoe. Lexical Issues in Natural Language Processing. In E. Klein and F. Veltman (eds.), *Natural Language and Speech,* pages 39–68, Springer-Verlag, 1991.

T. Briscoe, A. Copestake and B. Boguraev. Enjoy the paper: Lexical semantics via lexicology. In *Proceedings of the 13th Coling,* pages 42–47, Helsinki, 1990.

T. Briscoe, A. Copestake and V. de Paiva (eds.). *Default Inheritance in Unification based approaches to the Lexicon.* Cambridge University Press, New York, forthcoming.

N. Calzolari. The dictionary and the thesaurus can be combined. In M. W. Evens (ed.), *Relational models of the lexicon,* pages 75–96, Cambridge University Press, 1988.

B. Carpenter. Typed feature structures: Inheritance, (In)equality and Extensionality. In *Proceedings of the First International Workshop on Inheritance in Natural Language Processing,* pages 9–18, Tilburg, The Netherlands, 1990.

B. Carpenter. Skeptical and Credulous Default Unification with Applications to Templates and Inheritance. In T. Briscoe, A. Copestake and V. de Paiva (eds.), *Default Inheritance in Unification based approaches to the Lexicon,* CUP, New York, 1991, forthcoming.

J. Carroll and C. Grover. The derivation of a large computational lexicon for English from LDOCE. In B. Boguraev and T. Briscoe (eds.), *Computational lexicography for natural language processing,* pages 117–134, Longman, London, 1989.

J. Carroll. Lexical Database System: User Manual. Esprit BRA-3030 ACQUILEX deliverable no. 2.3.3(c), April 1990.

A. Copestake. An approach to building the hierarchical element of a lexical knowledge base from a machine readable dictionary. In *Proceedings of the First International Workshop on Inheritance in Natural Language Processing,* pages 19–29, Tilburg, The Netherlands, 1990.

A. Copestake. Default Unification in the LKB. In T. Briscoe, A. Copestake and V. de Paiva (eds.), *Default Inheritance in Unification based approaches to the Lexicon,* CUP, New York, 1991, forthcoming.

A. Copestake and T. Briscoe. Lexical Operations in a Unification Based Framework. In *Proceedings of the ACL SIGLEX Workshop on Lexical Semantics and Knowledge Representation,* pages 88–101, Berkeley, California, 1991.

A. Copestake, B. Jones, A. Sanfilippo, H. Rodriguez and P. Vossen. Multilingual lexical representation. ms University of Cambridge, Computer Laboratory, 1992.

W. Daelemans. Inheritance in Object-Oriented Natural Language Processing. In *Proceedings of the First International Workshop on Inheritance in Natural Language Processing,* pages 30–39, Tilburg, The Netherlands, 1990.

R. Evans and G. Gazdar (editors). The DATR papers. Cognitive Science Research Paper CSRP 139, School of Cognitive and Computing Sciences, University of Sussex, 1990.

P. Procter (editor). *Longman Dictionary of Contemporary English.* Longman, England, 1978.

J. Pustejovsky. Current issues in computational lexical semantics. In *Proceedings of the 4th European ACL,* pages xvii–xxv, Manchester, 1989.

J. Pustejovsky. The Generative Lexicon. *Computational Linguistics,* 17(4) 1991.

A. Sanfilippo. Grammatical Relations, Thematic Roles and Verb Semantics. PhD thesis, Centre for Cognitive Science, University of Edinburgh, 1990.

A. Sanfilippo and V. Poznanski. The Acquisition of Lexical Knowledge from Combined Machine-Readable Dictionary Sources. In *Proceedings of the 3rd Conference on Applied Natural Language Processing,* Trento, Italy, 1992.

P. Vossen. Converting Data from a Lexical Database to a Knowledge Base. ACQUILEX Working paper, No 27, 1991.

P. Vossen and A. Copestake. Untangling definition structure into knowledge representation. In T. Briscoe, A. Copestake and V. de Paiva (eds.), *Default Inheritance in Unification based approaches to the Lexicon,* CUP, New York, 1991, forthcoming.

Y. Wilks, D. Fass, C-M. Guo, J. McDonald, T. Plate T and B. Slator. A tractable machine dictionary as a resource for computational semantics. In B. Boguraev and T. Briscoe (eds.), *Computational lexicography for natural language processing,* pages 193–231, Longman, London, 1989.