Evaluation of lexical representation formalisms

Daelemans, W.M.P.; van der Linden, H.J.B.M.

TILBURG UNIVERSITY
KATHOLIEKE UNIVERSITEIT BRABANT

[redacted]

ITK MEMO

INSTITUTE FOR LANGUAGE TECHNOLOGY AND ARTIFICIAL INTELLIGENCE

INSTITUUT VOOR TAAL- EN KENNISTECHNOLOGIE

ITK Research Memo
february 1992

Evaluation of Lexical Representation Formalisms

Walter Daelemans &
Erik-Jan van der Linden

No. 14

We wish to thank Hans Boers with whom we worked on DATR. Furthermore, we thank Gosse Bouma, Koenraad de Smedt, other participants of the second CLIN meeing, and participants of the ITK/LIKE Colloquium for comments and discussion.

Evaluation of Lexical Representation Formalisms

Walter Daelemans and Erik-Jan van der Linden*

February 1992
(Draft, comments welcome)

Abstract

With the structure and organisation of the lexicon at the centre of attention in computational linguistics the last few years, we have witnessed the appearance of various proposals for lexical representation formalisms and mechanisms. We discuss formalism-internal and formalism-external criteria that allow us to compare and evaluate a number of these proposals (typed feature structures, DATR, the hierarchical lexicon, default unification, object-oriented representation). This evaluation results in a position with respect to the role of inheritance in the design of lexicons.

1 Introduction

The shift from the expression of linguistic generalizations in linguistic rules to their representation in the lexicon has led to a vast increase in the interest for the lexicon and the formal representation of lexical knowledge.[1] Unification-based approaches have introduced various formal devices (templates, default unification, typing etc.) to structure lexical data. Another line of research is formed by systems inspired by AI knowledge representation, that introduced the notion of *inheritance* to computational linguistics.[2]

The aim of this paper is to review developments in this area. We do not introduce a new formalism for the representation of lexical knowledge, but discuss a number of formalisms with acknowledged criteria in mind: *notational adequacy* and *expressivity*. An evaluation like this is important since various NLP projects are faced with a choice in favour of one of the formalisms in question, and has, to our knowledge, not been made elsewhere. Nebel and Smolka (to appear) only present a comparison of *formal* properties of what they refer to as *terminological representation languages* and *unification grammars*. The main result of our exercise will be a position with respect to the relation between unification and inheritance in lexicon design. Our slogan is *Inheritance before unification.*

2 Basic Functions and Evaluation Criteria

In AI, a fruitful distinction is made in the design of knowledge systems between a computational level (the level of representation and problem solving formalisms), and a knowledge level (implementation-independent aspects of the knowledge to be represented). Furthermore, it seems clear that these knowledge-level aspects (such as task analysis, inference patterns, domain models, problem-solving methods, and their interaction) determine the choice of architecture and formalisms used in the implementation of the knowledge system (Steels, 1989), and

*We wish to thank Hans Boers with whom we worked on DATR. Furthermore, we thank Gosse Bouma, Koenraad de Smedt, other participants of the second CLIN meeting, and participants of the ITK/LIKE Colloquium for comments and discussion.

[1] See for instance papers in the special issue on the lexicon of Computational Linguistics 1987; Boguraev and Briscoe 1989; Zernik 1989; van der Linden and van der Wouden 1990; Briscoe et al. 1992.

[2] See for instance papers in Daelemans and Gazdar, 1990, 1992.

not the other way round. The same applies to the design of computational lexicons (construed as lexical knowledge systems).[3]

From the point of view of performance, a task analysis of the lexicon in generation and parsing reveals the following basic functions, which any lexical representation formalism should facilitate as much as possible. In the remainder of this paper we will be concerned mainly with representation issues.

- **Representation.** Representation of morphological, syntactic, semantic and pragmatic information in such a format that it can be easily integrated, and used with grammar, parser, and generator.
- **Access.** Given underspecified information, lexical signs compatible with it must be retrieved. Given a lexical sign, information associated with this sign must be retrieved. Given a lexical sign, (information associated with) related signs must be retrieved. These are all instances of a generic *classification* task.
- **Acquisition.** Addition of new lexical signs to the lexicon must be possible while keeping consistency of the lexicon.

Classification as a generic task for lexicon access suggests a combination of abstraction, matching and refinement as a problem solving method, which in its turn suggests the use of *taxonomies* for the representation of lexical knowledge (compare Steels, o.c.). In other words, a knowledge level analysis (in terms of tasks and problem solving methods) may suggest important criteria for the evaluation of formalisms, in this case the basic relation in the lexicon.

Role of the formalism. We see a formalism as a bridge between theory (the knowledge level) and implementation (the computational level). There are different criteria that can be used to evaluate knowledge representation formalisms in general, and lexical representation formalisms in particular. Some of these are *formalism-internal*, and reflect the relation between formalism and implementation. Others are *formalism-external*, and reflect the relation between formalism and theory. The choice for taxonomies in lexical knowledge representation is an example of the application of a formalism-external criterion.

In current lexical research, formalism-internal criteria play a major role. If they are met by a formalism, it can be said to be *felicitous* from the point of view of mathematics and/or computer science.

- A formal declarative semantics should be provided.
- Formal declarative inference rules should be defined.
- Inference should be sound and complete.
- Inference should be computationally tractable.

The criteria we will focus on in this paper are *formalism-external* criteria[4]. They are, in our view, more important at the present stage of research in Computational Linguistics: mathematics and computer science have provided formal linguistics with a multitude of formal devices for the representation of linguistic data that match the criteria mentioned above, but less attention has been paid to the relation between (linguistic) theory and formalism.

- **Notational adequacy.** "Some kinds of notation seem to fit the sorts of facts one encounters in some domain; others, which may ultimately be equivalent in some sense to the former kinds, do not." (Gazdar, 1985). Applied to grammar formalisms, Shieber (1986:5) states that a formalism should represent linguistic phenomena "as linguists would wish to state them".
- **Expressivity.** Expressivity means that the formalism should allow for the expression of all linguistic facts, and only those. Depending on the opinion one has on which phenomena are linguistic and which are extra-linguistic, there is room for religious wars about expressivity.

[3] See van der Linden et al. 1990 for a comparable observation concerning the application of techniques from database design to the design of lexical databases.

[4] For these criteria see e.g. Whitelock (1987).

Expressivity of a formalism is concerned with *what* can be represented, notational adequacy with *how* it is represented. If these linguistic formalism-external criteria are met by a formalism it can be said to be *linguistically felicitous* [5]. Notice that we could also introduce *psychological felicity* this way. Another source of criteria which we will not discuss here, is *practical felicity*: *software engineering* considerations (parsimony, ease of maintenance, uniformity, modularity, and interaction (e.g. Daelemans et al. 1992) are not an issue here, but the use of inheritance hierarchies is motivated by such considerations.

Criteria may conflict. A model of the lexicon that is for instance linguistically felicitous removes as much redundancy as possible from the lexical representations, since one of the aims of linguistics is to provide an abstract and general description of language. Regularly inflected forms should not occur as stored lexical entries in the linguistic lexicon. A model that is psycholinguistically felicitous, on the other hand, may represent regularly inflected forms as such (Stemberger and McWhinney, 1986). The criteria of psychological realism and notational adequacy are thus in conflict. We will not pursue the old discussion concerning the role of cognitive psychology in AI, but concentrate on linguistic felicity.

3 Dimensions in hierarchical lexicon design

In this section, we discuss a number of dimensions in lexicon design, and for some of them we state what choice should be made given the criteria of the previous section. The main conclusion of this section, and the paper, will be that inheritance should be the basic relation between elements in the hierarchical lexicon, with unification playing a secondary role.

3.1 Basic Relation

In lexicalist grammars, there is a practical problem of removing redundancy in representation (feature structures associated with a single lexical sign can be hundreds of nodes big), and a theoretical problem of expressing the right generalizations. The non-redundant representation of lexical information, and the expression of generalizations can be accomplished by structuring the lexicon in the form of a hierarchy in which properties are shared between the elements that constitute the hierarchy. As we mentioned earlier, a taxonomic representation is also suggested by the main task of the lexicon in linguistic processing (i.e. classification).

In unification-based systems, the basic elements in the lexicon are signs (denoted here with capitals: X, Y, but mostly unnamed), consisting of features structures (feature = value) which denote the properties of the signs. The basic relation between signs is the subsumption relation ($\sqsubseteq$), from which unification is derived as an information combining operation (see Shieber, 1986).

(1) X
foo = 1

Y
foo = 1
bar = 2

$X \sqsubseteq Y$

In inheritance-based systems, the basic elements are classes or objects (X, Y; sometimes called types or prototypes) consisting of property-value relations (property = value). The basic relation between these classes is the inheritance relation, which can be interpreted as a generalization of more restricted relations.

Either a value (atomic or complex, i.e. a structured object in its own right) is provided for the property of an object (a), or a direction of where to look for the value (b-e). In (b), the value is equaled with the value of a different property within the same object, in (c) the value is equaled with the value for the same property in a different object, and in (d), the value is

[5] Although Shieber (1986) equals linguistic felicity with expressivity.

equaled with the value of a different property in a different object. In (e), all values of properties in object X are equaled to the values of the corresponding properties in Y (α ranges over all properties). This corresponds to the well-known IS-A inheritance relation (X IS-A Y). Parts of expressions which are left implicit in the syntax of most formalisms are between brackets.

(2) a. X
foo = 1

Y
foo = 2
bar = 3

b. X
foo = 1
bar → (foo of) X

c. X
foo → (foo of) Y

d. X
foo → bar of Y

e. X
(α) → (α of) Y

3.2 Recursive Structure

It should be possible to allow lexical signs to refer to other lexical signs as values in their information structure. We want to be able to express *paths* of properties or features, as in the following (where uppercase symbols are structured lexical signs or classes).

(3) WORK
type = REGULAR-VERB
past-part = WORKED

WORKED
type = VERB-FORM
tense = past
person = 1 ∧ 2 ∧ 3
number = sing ∧ plur

INTRANSITIVE-VERB
category = (np\(np \s))

AUXILARY
argument = INTRANSITIVE-VERB
result = INTRANSITIVE-VERB

This allows us to build complex paths while querying the lexicon, e.g. *What is the person of the past-part of WORK*, or to describe the relation between argument category and result category of the categorial representation for an auxilary. Similarly for syntax (e.g. we can have complex NP's as values in subject and object slots), or for semantics (e.g. we can have structured semantic types as values in the semantics attribute). Although at least the necessity of this type of expressivity seems to be self-evident, we will see that it is not unproblematic in some lexical representation formalisms.

In what follows we will review a number of design principles for hierarchical lexicons.[6]

3.3 Multiple and Single Inheritance

In single inheritance, each element has at most one parent element it inherits information from. In multiple inheritance it is possible to inherit from more than one parent. Inheritance must be multiple in order to achieve notational adequacy, that is, to avoid redundant descriptions and to improve modularity. In the definition of lexicons, we want to separate morphological, syntactic and semantic information in different taxonomies, and integrate this information in lexical signs through multiple inheritance, because the generalizations we want to express in the different parts of the lexicon will suggest differently organized hierarchies.

For instance, *admire* inherits its syntactic information from the class TRANSITIVE-VERB, its semantic information from PSYCHOLOGICAL-VERB, and its morphological information from REGULAR-VERB. Notational adequacy would suffer if we tried to cram all these classes into the same taxonomy.

When an element inherits from more than one other element it is possible that conflicts may arise, however. There are basically two types of solution that have been suggested for this problem: *orthogonal multiple inheritance* and *prioritized multiple inheritance*.

3.3.1 Orthogonal Multiple Inheritance

This solution, which is in line with the motivation for the use of multiple inheritance as a means of partitioning the lexical database, suggests that information inherited from different classes should be non-conflicting; no single property can be inherited from more than one parent class. This principle of *orthogonal inheritance* is present in work by Flickinger (1987:61), in the object-oriented morphology work of Daelemans (1987b:50-53), and in the 'parent psorts do not conflict' principle of Copestake (1991:2).

3.3.2 Prioritised Multiple Inheritance

In this view of multiple inheritance, one or more properties may be inherited from more than one parent, but the parents are ordered (in a class precedence list). Of two parents with conflicting values for this property in this list, the first wins, and *shadows* the value of the property in the other. This *prioritised multiple inheritance* (present in languages such as FLAVORS and CORBIT) has been used in object-oriented NLP extensively (e.g. De Smedt 1984, Daelemans 1989). Recently it has been granted theoretical linguistic status in the work of Carpenter (1991) and Russell et al. (1992), which is based on the class precedence list computation method of CLOS (Steele, 1990). It is an open question which algorithm can best be used to compute class precedence: depth first, breadth first, topological ordering, inferential distance (Touretzky, 1986) or others. In Daelemans (1990) it is suggested that the class precedence list can be dynamically computed on the basis of contextual cues during processing.

3.4 Monotonic and Non-monotonic Inheritance

In monotonic inheritance, each element inherits *all* properties associated with its parent. To be able to express regularities, sub-regularities and exceptions within a single hierarchy, we have to abandon monotonicity and introduce the principle that properties attached to an element take precedence over those inherited from a parent. This is called non-monotonic inheritance, or *default inheritance*[7]. Although in principle, default inheritance could be avoided by using multiple non-monotonic inheritance, notational adequacy demands the capability of stating

[6] See Touretzky and Thomason, 1987 for a more general overview of inheritance mechanisms. A number of distinctions presented there and elsewhere in the literature will not be discussed here: class-based inheritance versus prototype-based inheritance, homogeneous versus heterogeneous inheritance, unipolar versus bipolar inheritance, and cyclicity.

[7] See Dörre et al. for an overview of formalism-internal considerations concerning defaults.

class definitions as being of a certain parent class, with some additional (possibly conflicting) information.[8]

One of the earliest illustrations of the descriptive power of default inheritance is the following treatment of Dutch verbs by De Smedt (1984).

```
         WEAK VERB
        /      past part     = ge+root+t/d
       /       past sing     = root+t/de
      /        pres sing 3   = root+t
     /         ...
    /                  \
WERKEN            MIXED VERB
                /        past part  = ge+root+en
               /             \
              /               \
        BAKKEN               STRONG VERB
                           /     past sing = +Vowel Change
                          /
                     LOPEN
```

The expressiveness of default inheritance has been used in work in HPSG (Flickinger, Pollard and Wasow, 1985; Flickinger, 1987), Word Grammar (Hudson, 1984, 1990), and object-oriented NLP (Daelemans, 1987b; De Smedt, 1990), and has been the prime motivation for work on default unification (Bouma, 1992; Carpenter, 1991).

The (mostly implicit) consensus view has been that as far as notational adequacy is concerned, default inheritance hierarchies should be used to encode dimensions of regularity, markedness, and productivity within a single hierarchy (see Daelemans et al., 1992).

3.5 Explicit coding of non-default information

It is possible to code non-default information explicitly, or to leave it implicit, that is, to let the information the child carries prevail over the information of the parent. The latter principle is an acknowledged principle in knowledge representation and (computational) linguistics, and goes under the name of *blocking*, *elsewhere condition* or *principle of priority to the particular*. The difference between the figure below, and the one presented above, is the explicit coding of defeasability of the *past part* and *part sing* information with the use of a overwriting operator, *!*.

```
         WEAK VERB
        /      past part     = ge+root+t/d
       /       past sing     = root+t/de
      /        pres sing 3   = root+t
     /         ...
    /                  \
WERKEN            MIXED VERB
                /        ! past part   = ge+root+en
               /             \
              /               \
        BAKKEN               STRONG VERB
                           /     ! past sing = +Vowel Change
                          /
                     LOPEN
```

[8] See Daelemans et al. 1992 for a discussion of this point.

In extensions of unification-based frameworks, one finds explicit coding of overwriting (Bouma 1992; Carpenter 1991)[9].

There are two problems with the explicit coding of non-defaults. Firstly, the lexicon designer must indicate explicitly what information is default and what information is non-default. One could say that explicit coding represents such statements as "X is a Y, *but* its c-property is d", and implicit coding represents such statements as "X is a Y, with d as its c-property". Note that in the default inheritance mechanisms discussed earlier, the most specific information takes precedence automatically. It is questionable whether the first kind of statement matches the demand for expressivity. Secondly, a problem occurs when several pieces of information in the hierarchy are marked as non-default information. In default unification all default and non-default information must be unified separately before it is combined by default unification. This implies that different pieces of non-default information may not be conflicting, which makes it impossible to define hierarchies with exceptions of exceptions.

(4) BIRD
Can-fly = Yes

PENGUIN
isa BIRD
! Can-fly = No

TWEETY
isa PENGUIN
! Can-fly = Yes

A set-theoretic approach. A set-theoretic approach makes the differences between the approach which necessitates explicit overwriting and other approaches clearer. The subsumption relation is comparable to a subset-superset relation in set-theoretic terms. If we take num = sing to denote a set the members of which have the property num = sign, and person = 1 to denote a set the members of which have the property person = 1, then num = sing ∩ person = 1 denotes the intersection of these sets. If a mothernode occurs in this intersection then the nodes subsumed by the mother should be members of the intersection as well. In approaches in which inheritance is the principle relation between signs, all properties are in principle defeasible, and so a mother in num = sing ∩ person = 1, may have children which are in some subset of the union of the sets denoted by the properties of the mother. For instance a node for which num = plural and person = 1 hold may be children of the mother. If in the unification approach a node deviates with respect to certain properties from the node, this should thus be coded explicitly.

Explicit overwriting in Word Grammar. Default inheritance implies a form of *blocking*: the computation of regular forms is shielded by the presence of irregular forms at a more specific level in the default inheritance hierarchy (this was even one of the prime motivations for introducing it into lexical description). However, in some cases (e.g. the English plural), we want to derive both the exceptional and the regular form (*hoof*: *hooves*, *hoofs*).[10] To solve this problem, Fraser and Hudson (1992) introduce a notion of inheritance in which exceptions do not automatically override inherited defaults, the default has to be explicitly negated if it has to be suppressed.

[9] Note that the explicit coding of defeasability in Veltman 1990:31, and Touretzky 1986 is a property of the information that is overwriten, and thus differs from the notion of explicit non-default information here.

[10] The same applies to the verb *prove* from which two participles can be derived: *proven* and *proved*.

(5) NOUN
Plural = Root + s

FOOT
Isa NOUN
NOT Plural = Root + s
Plural = Feet

HOOF
Isa NOUN
Plural = HOOVES

Note that this way a device has to be added for the exceptional case where we have two plural forms. We loose the more notationally adequate automatic blocking present in other inheritance systems. In default unification we have to know which information is not default, in Word Grammar we have to know which default information is overridden. We would favour a device which explicitely states that information is *added*, or is a variant, which is notationally more adequate in our view. This could be accomplished by means of an operator **VARIANT**, or by means of the variant sets introduced by Russell et al. 1992. Below we present a tentative representation.

(6) NOUN
Plural = Root + s

FOOT
Isa NOUN
Plural = Feet

HOOF
Isa NOUN
Variant Plural = HOOVES

3.6 Conclusion

The conclusion from the present section is that a formalism that meets the formalism-external criteria set out at the beginning should at least have the following properties.

- Recursive structuring (path formation).
- Integration of knowledge from multiple sources (multiple inheritance/subsumption).
- Default reasoning (non-monotonic inheritance/subsumption).
- Implicit blocking (no explicit coding of default information).

The choice of a basic relation is left implicit here, and will emerge from the discussion of the state of the art in the next section. With respect to multiple default inheritance, we conclude that *orthogonal* multiple default inheritance is at this stage the best solution for conflicts. With unrestricted multiple inheritance, the advantages in general don't weigh up against the formal intricacies of dealing with conflicting inherited information. More research remains to be done to evaluate the adequacy of different proposals for *prioritised* multiple inheritance.

4 State of the art

In this section we discuss the state of the art in lexical representation formalisms, and use the criteria and design decisions from the previous sections to evaluate them.

4.1 Unification-based formalisms

Unification as used in currently popular unification-based grammar (UBG) formalisms is by definition a monotonic operation. Information in the templates of PATR (Shieber, 1986), e.g. is combined by means of unification, and the operation is equivalent in expressive power to information combination by means of (multiple) monotonic inheritance. This monotonicity is at the basis of some of the acknowledged advantages of UBG, e.g. order independence.

Part of the study of unification-based formalisms has been directed at extending the unification-based machinery with tools for the representation of lexical knowledge. After the introduction of such concepts as templates and overwriting in PATR, two other concepts have become of importance: *typing* and *default unification.*[11]

4.1.1 Default Unification

Bouma (1992) presents an asymmetric operation of *default unification* for (untyped) features structures. The operation results in combining as much default information as possible with the non-default (or strict) information. The basic relation in the lexicon is a unification relation.

In order to distinguish default from non-default information Bouma uses a unary type constructor *!* with feature-value pairs in its domain. Note that use of such a type constructor is obligatory: in case of non-default information there exists a conflict between the information provided by the mother and the information provided by the daughter, which would lead to subsumption/unification failure. Combining the non-default information by means of unification would fail in this case before default unification can be applied. However, as we noted before, explicit coding of non-default information is problematic.

In Krieger and Nerbonne (1991) it is proposed that feature structures are used in lexical representation and for inflectional and derivational morphology. Again, the basic relation is subsumption, and Bouma's default unification is acclaimed as being sufficient for representing default inheritance.

4.1.2 Typed Feature Structures

Multiple *monotonic* inheritance hierachies can be found in a great deal of work in unification-based grammar. In HPSG (Pollard and Sag, 1987), the lexicon is treated as a monotonic multiple inheritance hierarchy. Other examples are Emele and Zajac (1992) and Emele et al. (1990), sorts in Unification Categorial Grammar (Moens et al. 1989) and in the CLE project (Alshawi et al. 1989), and the proposal of Nebel and Smolka (1991). In Typed Feature Structures (see for instance Zajac, 1992) types are represented in a hierarchy with subsumption as the basic relation. Although Zajac calls this an inheritance network, it should thus rather be called a *subsumption network*. It is not suprising that under this strict definition of inheritance as subsumption, no notion of default inheritance is possible: "A subtype inherits all constraints of its supertype monotonically". On the basis of expressive adequacy this restriction should be rejected.

Carpenter 1991. Carpenter (1991) presents a formal system in which inheritance precedes unification: feature structures may have a *default unification relation*, but the relation between elements in the lexicon is basically an *isa* relation. Carpenter, however, uses notions of *templatic inheritance* and *default inheritance* which have the same flaws as Bouma's (1992) system: overwriting and defaults have to be coded *explicitly* by marking a feature-value structure with a one-place operator *?* or *!* (templatic inheritance), or a one-place operator *strict* or *default* (default inheritance). It is not surprising that Carpenter faces the problem of order-sensitivity for the resolution of conflicting defaults.

[11] Gazdar et al. 1988 present a general unification-based formalism for the representation of category structures which is limited to syntactic information. Dörre et al. present an overview of extensions of unification-based formalisms.

4.2 DATR

DATR (Evans and Gazdar, 1989a,b, 1990) is a non-monotonic inheritance formalism for lexical knowledge representation that is intended to be notationally adequate and formally and computationally well-behaved. Lexical knowledge is expressed in terms of path equations.

The following sample DATR lexical theory for instance, expresses that the verb node is associated with syntactic category *verb* and syntactic type *main*, and that morphologically (by default inheritance) *all* past forms consist of a sequence whose first element is the root and whose second element is the suffix +ed. In the absence of any more specific (longer) paths, all extensions of <mor past> in the context of VERB or descendents of VERB like AUX will inherit the same value. As far as the present form is concerned: for all extensions of <mor pres tense> except explicitly defined more specific ones like <mor pres tense sing three> the value "<mor root>" is inferred. Quoted paths are evaluated *globally*, i.e. in this example, "<mor root>" refers to the value for <mor root> in the node description the query started from (possibly at a lower level in the hierarchy). Paths that are not quoted are evaluated in the context of the node with which they are associated. Notice that as far as notational adequacy is concerned, the *normal* case is explicitly marked in DATR syntax (with quotes), while the marked case (global inheritance) is unmarked.

```
VERB: <syn cat> == V
      <syn type> == Main
      <mor past> == ("<mor root>" +ed)
      <mor pres tense> == "<mor root>"
      <mor pres tense sing three> == ("<mor root>" +s)
      <mor pres participle> == ("<mor root>" +ing)

AUX: <> == VERB
     <syn type> == aux
```

There are more limitations to the notational adequacy of DATR, however.

- Apart from supporting wanted inferences by default inheritance through path extension, DATR also happily infers nonsense like <mor past pres future> = <"<mor root>" +ed>. To prevent some of the extensions to be inferred, so-called *ugly objects* should be defined. Taken together with the atomic attribute interpretation of DATR paths to be discussed shortly, this means that DATR default inheritance is in fact merely an instance of *prefix matching* of attributes. From this respect the expressivity of DATR is not adequate.
- The type of "multiple" inheritance used in DATR is a kind of mixin inheritance: inheritance pointers can only be specified between Node-Path combinations, not between nodes (objects). Single inheritance is represented as

  ```
  X:<> == Y:<>.
  ```

  This makes it possible to ascertain that all multiple inheritance in DATR is orthogonal. Evans et al. (1992) explicitly discuss multiple inheritance in DATR. Although they present a DATR-like pseudo-formalism, in which multiple prioritised inheritance could be expressed *as such*,

  ```
  ABC:<> == A,B,C
  ```

  they do not incorporate this in DATR. Multiple inheritance is claimed to be expressible, but this is only possible through a detour, and with the use of rather intricate DATR-constructions. (As DATR is a powerful general-purpose representation language it is not surprising that this is possible)[12]. Linguistic felicity would demand the expression that ABC is an A, a B and a C in one simple statement, however. The prioritised multiple inheritance discussed by Evans et al. (1992) applies to DATR-paths, not to inheritance from different objects.

[12] For the more complex syntactic possibilities of DATR, no formal declarative semantics has been proposed yet.

- It is not possible in DATR to have complex structured objects as "values" (remember that only equations of node-path pairs can be expressed). This is unfortunate because, as we pointed out earlier, we want to be able to express this kind of recursion in the lexicon. Related to this, neither nodes nor theories in DATR correspond to lexical entities.
- Apart from these issues, there are a few points with respect to formalism-internal criteria we would like to mention. The similarity of DATR paths to PATR paths is superficial, their semantics is very different. In fact, the DATR paths could be better described as atomic attributes (they do not correspond with a recursive structure, and their only function is to support prefix matching). Despite their superficially similar syntax, the integration of DATR lexical theories with UBG-usable feature structure definitions is far from trivial (although possible using the more complex syntactic possibilities mentioned earlier). All proposed solutions seem to require the off-line expansion of all lexical material (as opposed to on-line or lazy inference). How to translate UBG-specific mechanisms like reentrancies and disjunction into DATR is at present an unsolved question.

4.3 Object-Oriented Natural Language Processing

In object-oriented NLP (De Smedt, 1984, 1990; Daelemans, 1987b,88), ideas from object-oriented languages like polymorphism, inheritance and encapsulation, are transported to linguistic modeling. (See Daelemans, 1989 for an overview and related ideas in AI, linguistics and computational linguistics).

As a lexical representation language (as proposed in Daelemans, 1987a), object-oriented representation allows the definition of single and multiple inheritance hierarchies. While most existing work presupposes a form of prioritised inheritance (enforced by a default search strategy built into the definition of the language), it is easy to guard orthogonality in multiple inheritance. Also, while most existing work makes use of procedural *methods*, a declarative regime can be enforced. (Typed) feature structures can straightforwardly be implemented as objects (Daelemans et al. 1991), allowing the combination of unification and inheritance in a single system. Due to the fact that lexical entities have a straightforward implementation as structured objects, the approach seems to be intuitively clearer.

From the point of view of the criteria discussed here, object oriented frameworks provide notational adequacy, but with respect to expressivity, unconstrained object oriented frameworks are too powerful. Also, one important difference between this framework and the other formalisms in this paper, is that there exists no formal semantics for object-oriented formalisms in general (although proposals exist for significant subsets).

4.4 Hybrid Systems

Apart from the object-oriented approach discussed in the previous section, a number of other systems combine feature-based and inheritance relations (Carpenter 1991, section 5; Russell et al. 1992; Copestake 1991). In these systems, the basic relation between signs is the subclass-superclass relation. Notions like *default extension*, *superclass extension* and *global extension* mediate between the inheritance relation and the unifiability of information of the classes. For these hybrid systems not all formalism-internal demands have been fulfilled yet.

5 Conclusion

The conclusion we draw from the demands on lexical representation formalisms and the state of the art is that lexical representation formalisms should allow for *orthogonal*, *multiple*, *not explicitely coded*, *default inheritance* and for *recursive structures*. Our position with respect to existing formalisms is the following.

- Formalisms that take unification as the basic relation and their extensions (typed features structures and default unification) do not allow for a proper way to represent nonmonotonic inheritance which is necessary from the point of view of expressivity.

- DATR does not allow for multiple inheritance in a notationally adequate fashion, and is too expressive as concerns path equations.
- Object-oriented formalisms are too expressive since they allow for the use of the complete expressivity of programming languages.
- Hybrid Systems are the most promising since they allow for inheritance as a basic relation between signs, and its formulation in terms of unification.

Future research Although we do not agree with his implementation of it, we agree with the following quote from Zajac (1992:3). "A linguistic formalism should be an object-oriented logic formalism." It remains to be seen what will turn out to be the best approach to accomplish this. One line could be to pursue the limitation of object oriented frameworks. One of the interesting devices that should be preserved is the concept of multimethods (Daelemans, 1990). In order to provide a formal semantics for OO frameworks, it should be noted that it is clear from the work of Touretzky (1986) and Veltman (1990) that a modeltheoretic formal semantics of a nonmonotonic framework requires an intensional approach, a opposed to the extensional semantics that can for instance be found in the work of Carpenter (1991).

Another line of research would be to present inheritance in terms of logical type constructors as in van der Linden (1992), although one could say that from the point of view of type theory these operators should in principle be represented on a meta-level.

References

Hiyan Alshawi, David Carter, Johan van Eijck, Robert Moore, Douglas Moran, Fernando Pereira, Stephen Pulman, and Arnold Smith. 'Final Report: Core Language Engine', Technical Report, Project No. 2989, SRI, Cambridge, 1989.

Branimir Boguraev and Ted Briscoe, (eds.) *Computational Lexicography for Natural Language Processing.* London: Longman, 1989.

Gosse Bouma, 'Feature structures and nonmonotonicity', *Computational Linguistics*, vol. 18, no. 2, 1992, forthcoming.

Ted Briscoe, Ann Copestake & Valeria de Paiva (eds), *Default Inheritance in the Lexicon*, Cambridge University Press, Cambridge, 1992.

Robert Carpenter, 'Skeptical and Credulous Default Unification with Applications to Templates and Inheritance.' In: Ted Briscoe, Ann Copestake & Valeria de Paiva (eds), *Proceedings Acquilex Workshop on Deafaults in the Lexicon.*, 1991.

Anne Copestake, 'Defaults in the LRL.' In: Ted Briscoe, Ann Copestake & Valeria de Paiva (eds), *Proceedings Acquilex Workshop on Deafaults in the Lexicon.*, 1991.

Walter Daelemans. 'A Tool for the Automatic Creation, Extension and Updating of Lexical Knowledge Bases.' *Proceedings of the Third ACL European Chapter Conference*, Copenhagen, 70-74, 1987a.

Walter Daelemans. *Studies in Language Technology: An Object-Oriented Computer Model of Morphophonological Aspects of Dutch*, University of Leuven, PhD dissertation, 1987b.

Walter Daelemans. 'A model of Dutch morphophonology and its applications', *AI Communications*, vol. 1, no. 2, pp. 18-25, 1988.

Walter Daelemans. 'Object-oriented Hierarchical Lexicons.' *Proceedings IJCAI'89 Workshop on Object-oriented Programming in AI*, Detroit, 1989.

Walter Daelemans. 'Inheritance in Object-oriented Natural Language Processing.' In: Daelemans, W. and G. Gazdar (eds.) *Inheritance in Natural Language Processing: Workshop Proceedings* Tilburg: ITK, ITK Proceedings 90/1, 1990, p.30-38.

Walter Daelemans and Gerald Gazdar (eds.) *Inheritance in Natural Language Processing: Workshop Proceedings* Tilburg: ITK, ITK Proceedings 90/1, 1990, 120p.

Walter Daelemans, Koenraad De Smedt and Josje de Graaf. 'Default Inheritance in an object-oriented representation of linguistic categories'. ITK Research Report 31, 1991.

Walter Daelemans and Gerald Gazdar (eds.) Inheritance in Natural Language Processing. Special Issue of Computational Linguistics. Vol 18 (2), 1992, forthcoming.

Walter Daelemans, Koenraad De Smedt & Gerald Gazdar. 'Inheritance in Natural Language Processing.' *Computational Linguistics* Vol 18 (2), 1992.

Koenraad De Smedt, 'Using object-oriented knowledge representation techniques in morphology and syntax programming', *ECAI-84*, 181-184, 1984.

Koenraad De Smedt, *Incremental Sentence Generation: a Computer Model of Grammatical Encoding.* NICI Technical Report 90-01, Nijmegen, 1990.

Martin C. Emele and Rémi Zajac, 'Typed unification grammars', COLING-90, pp. 293-298, 1990.

Martin C. Emele, Ulrich Heid, Stefan Momma, and Rémi Zajac, 'Organizing linguistic knowledge for multilingual generation', *COLING-90*, pp. 102-107, 1990.

Roger Evans and Gerald Gazdar, 'Inference in DATR', *ACL Proceedings*, 4th European Conference, pp. 66-71, 1989a.

Roger Evans and Gerald Gazdar, 'The semantics of DATR', in *Proceedings of the Seventh Conference of the Society for the Study of Artificial Intelligence and Simulation of Behaviour*, ed. Anthony G. Cohn, pp. 79-87, Pitman/Morgan Kaufmann, London, 1989b.

Roger Evans and Gerald Gazdar, *The DATR Papers, Volume 1*, Cognitive Science Research Paper CSRP 139, University of Sussex, Brighton, 1990.

Roger Evans, Gerald Gazdar, and Lionel Moser, 'Prioritised multiple inheritance in DATR', in *Default Inheritance in the Lexicon*, Ted Briscoe, Ann Copestake & Valeria de Paiva (eds), Cambridge University Press, Cambridge, 1992.

Daniel P. Flickinger, *Lexical Rules in the Hierarchical Lexicon*, Stanford University, PhD dissertation, 1987.

Daniel P. Flickinger, Carl J. Pollard, and Thomas Wasow, 'Structure-sharing in lexical representation', *ACL Proceedings*, 23rd Annual Meeting, 262-267, 1985.

Norman M. Fraser and Richard A. Hudson, 'Inheritance in Word Grammar', *Computational Linguistics*, vol. 18, no. 2, 1992.

Gerald Gazdar, 'Computational Tools for Doing Linguistics: Introduction', *Linguistics* 23, 185-187, 1985.

Gerald Gazdar, Geoffrey Pullum, Bob Carpenter, Ewan Klein, Thomas Hukari, and Robert Levine 'Category Structure', *Computational Linguistics* 14, pp. 1-19, 1988.

Richard A. Hudson, *Word Grammar*, Blackwell, Oxford, 1984.

Richard A. Hudson, *English Word Grammar*, Blackwell, Oxford, 1990.

Hans-Ulrich Krieger and John Nerbonne, 'Feature-based Inheritance Networks for Computational Lexicons.' *Proceedings of the Acquilex Workshop on Default Inheritance*, 1991.

Erik-Jan van der Linden, 'Incremental processing and the hierarchical lexicon', *Computational Linguistics*, vol. 18, no. 2, 1992.

Erik-Jan van der Linden, Sjaak Brinkkemper, Koenraad De Smedt, Pauline van Boven, and Mieke van der Linden, 'The Representation of Lexical Objects', In: Magay, T., and Zigány, J., (Eds.) *BudaLEX '88 proceedings : papers from the 3rd International EURALEX congress*, Budapest, 4-9 September 1988. Budapest: Akadémiai Kiadó, 1990.

Erik-Jan van der Linden & Ton van der Wouden, 'Computer and Lexicon.' In: Van der Wouden, T. & Van der Linden, E. (Eds.) *Computer and Lexicon.* Special issue of *Tabu*, 20, 3, p. 61-74, 1990 (Dutch).

Marc Moens, Jo Calder, Ewan Klein, Mike Reape, and Henk Zeevat, 'Expressing generalizations in unification-based grammar formalisms,' *ACL Proceedings*, 4th European Conference, 174-181, 1989.

Bernhard Nebel and Gert Smolka, 'Representation and Reasoning with Attributive Descriptions.' In: Bläsius et al. *Sorts and Types in Artificial Intelligence*, 1992.

Carl Pollard and Ivan A. Sag, *Information-Based Syntax and Semantics, Volume 1*, CSLI/Chicago University Press, Stanford/Chicago, 1987.

Graham Russell, Afzal Ballim, John Carroll, and Susan Warwick-Armstrong, 'A practical approach to multiple default inheritance for unification-based lexicons,' *Computational Linguistics*, vol. 18, no. 2, 1992.

Stuart M. Shieber, *An Introduction to Unification-Based Approaches to Grammar*, University of Chicago Press, Chicago, 1986.

Guy Steele, *Common Lisp The Language*, Digital: Digital Press, 1990.

Luc Steels, 'Components of Expertise.' AI-MEMO 89-2, 1989.

Joseph Stemberger and Brian MacWhinney 'Frequency and the lexical storage of regularly inflected forms', *Memory and Cognition* 14, pp.17-26, 1986.

David S. Touretzky, *The Mathematics of Inheritance Systems*, Pitman/Morgan Kaufmann, London/Los Altos, 1986.

David Touretzky & Richmond Thomason, 'A clash of intuitions: the current state of nonmonotonic inheritance systems.' In *Proceedings od IJCAI-87*, 676-482, 1987.

Frank Veltman, 'Defaults in Update Semantics' In: Hans Kamp (ed.) *Conditionals, Defaults and Belief revision.* Dyana Deliverable R2.5.A, 1990.

Jochen Dörre, Andreas Eisele, Jürgen Wedekind, Jo Calder and Mike Reape, 'A survey of Linguistically Motivated Extensions of Unification-Based Formalisms' Dyana Deliverable R3.1.A.

Peter Whitelock, Mary McGee Wood, Harold L. Somers, Rod L. Johnson, and Paul Bennett, *Linguistic Theory & Computer Applications*, Academic Press, London, 1987.

Rémi Zajac, 'Inheritance and constraint-based grammar formalisms', *Computational Linguistics*, vol. 18, no. 2, 1992.

Uri Zernik (Ed.) *Proceedings of the First International Lexical Acquisition Workshop.* Detroit, Michigan, 1989.

ITK: P.O. BOX 90153 5000 LE TILBURG THE NETHERLANDS