Inference in DATR

Roger Evans & Gerald Gazdar
School of Cognitive and Computing Sciences
University of Sussex, BRIGHTON BN1 9QN

Abstract

DATR is a declarative language for representing a restricted class of inheritance networks, permitting both multiple and default inheritance. The principal intended area of application is the representation of lexical entries for natural language processing, and we use examples from this domain throughout. In this paper we present the syntax and inference mechanisms for the language. The goal of the **DATR** enterprise is the design of a simple language that (i) has the necessary expressive power to encode the lexical entries presupposed by contemporary work in the unification grammar tradition, (ii) can express all the evident generalizations about such entries, (iii) has an explicit theory of inference, (iv) is computationally tractable, and (v) has an explicit declarative semantics. The present paper is primarily concerned with (iii), though the examples used may hint at our strategy in respect of (i) and (ii).

1 Introduction

Inheritance networks ("semantic nets") provide an intuitively appealing way of thinking about the representation of various kinds of knowledge. This fact has not gone unnoticed by a number of researchers working on lexical knowledge representation, *e.g.* de Smedt (1984), Flickinger et al. (1985), Calder & te Lindert (1987), Daelemans (1987a,1987b), Gazdar (1987) and Calder (1989). However, many such networks have been realized in the context of programming systems or programming languages that leave their precise meaning unclear. In the light of Brachman (1985), Etherington (1988) and much other recent work, it has become apparent that the formal properties of notations intended to represent inheritance are highly problematic. Although not discussed here, **DATR** has a formal semantics (Evans & Gazdar 1989) for which some completeness and soundness results have been derived. These results, and others (on complexity, for example) will be provided in a subsequent paper. There are several prototype computational implementations of the language, and non-trivial lexicon fragments for English, German and Latin have been developed and tested.

2 Syntax

The syntax of **DATR**, especially the use of value terminated attribute trees to encode information derives from **PATR** (Shieber 1986). The language consists of strings of symbols drawn from the set SYM = {:, ", ., =, ==, <, >, (,)} and the sets ATOM and NODE, all of which are disjoint.

A string is in **DATR**, (with respect to given sets ATOM of [atom]s and NODE of [node]s) iff it is a [sentence] as defined by the following set of rules:

```
[sentence] ::=  [node]:[path] == [lvalue].
                | [node]:[path] = [value].

[lvalue]   ::=  [latom] | ( [lseq] )
[gvalue]   ::=  [gatom] | ( [gseq] )
[value]    ::=  [atom]  | ( [seq] )

[latom]    ::=  [desc] | [gatom]
[gatom]    ::=  "[desc]" | [atom]
```

```
[desc]    ::=   [node] | [lpath]
                | [node]:[lpath]

[lseq]    ::=   [gseq] | [lseq] [desc] [lseq]
[gseq]    ::=   [seq] | [gseq] "[desc]" [gseq]
[seq]     ::=   ε | [value] [seq]

[lpath]   ::=   < [laseq] >
[path]    ::=   < [aseq] >

[laseq]   ::=   ε | [latom] [laseq]
[aseq]    ::=   ε | [atom] [aseq]
```

There are two kinds of sentence, those containing '==' and those containing '='. Both kinds have on their left-hand side a ***node:path*** specification, where a *path* is a sequence of atoms enclosed in <...>. Pragmatically, the '==' sentences are intended for defining the network, whilst the '=' statements express the values at individual nodes. Put another way, the former provide the database definition language whilst the latter provide the query language: the useful premises will standardly all be '==' statements, whilst the interesting theorems will standardly all be '=' statements (though the language itself also allows the former to be derived as theorems and the latter to be used as premises). In view of this distinction, we shall sometimes refer to '==' sentences as **definitional** and '=' sentences as **extensional**. Throughout the examples in this paper, we shall use **bold** for nodes and roman for atoms. ***Bold italic*** and *italic* will be used for corresponding meta-notational variables. Variables such as *N*, *P*, *L*, *G* and *V* will be assumed to be typed (as nodes, paths, lvalues, gvalues and values respectively). We shall sometimes refer to atoms occurring in paths as **attributes**.

The right-hand sides of extensional sentences are values, that is, simple atoms or lists of atoms/nested lists enclosed in (...). Lists are provided to allow the components of complex values to be specified independently (inherited from different places, for example). As an example, the following sentences might be derivable from a lexical entry for English 'be':

Be:<pres tense sing one> = am.
Be:<pres participle> = (concat be ing).

Likewise, the following for German 'Buch':

Buch: <sing> = Buch.
Buch: <plur> = (concat (umlaut Buch) er).

Values are the principal 'results' of a **DATR** description: the most typical operation is to determine the value associated (by an extensional sentence) with some node/path pair.

The right-hand sides of definitional sentences are lvalues, which can be simple atoms, inheritance descriptors (quoted or unquoted), or lists of lvalues. An atom is primitive, an inheritance descriptor specifies where the required value can be inherited from, and lists allow arbitrary structures to be built as values. Inheritance descriptors come in several forms with two dimensions of variation. The unquoted/quoted distinction specifies whether the inheritance context is local (the most recent context employed) or global (the initial context employed). Once the context is established, the descriptor specifies a new node, a new lpath, or both to be used to determine the inherited value. For example, the following sentences might be found in a description of a lexicon for English:

EN_MOR: <> == **VERB.**
EN_MOR:
<past participle> == (concat "<root>" en).
Take: <> == **EN_MOR.**
Take: <root> == take.

Finally an lpath is a path made up of lvalues, that is, elements which themselves may need evaluation, as in this example:

Adjective:
<form> == <"<gen>" "<num>" "<case>">.

We adopt the following abbreviation convention for sets of sentences about a single node:

N: *P1* == *L1*
P2 == *L2*
...
Pn == *Ln*.

abbreviates:

N: *P1* == *L1*.
N: *P2* == *L2*.
...
N: *Pn* == *Ln*.

and

N: *P1* = *V1*
P2 = *V2*
...
Pn = *Vn*.

abbreviates:

N: *P1* = *V1*.
N: *P2* = *V2*.
...
N: *Pn* = *Vn*.

Thus the 'take' example given above could appear, in abbreviated form, as follows:

EN_MOR:
<> == **VERB**
<past participle> == (concat "<root>" en).
Take:
<> == **EN_MOR.**
<root> == take.

3 Rule-based inference

DATR has seven syntactic rules of inference falling into three groups. The first rule just provides us with a trivial route from definitional to extensional sentences:

(I) $\dfrac{N{:}P == V.}{N{:}P = V.}$

For example, from:

VERB: <past> == ed.

one can infer:

VERB: <past> = ed.

Note that *V* must be a value (not an lvalue) here, otherwise the consequent would not be well-formed.

The next three rules implement local inheritance of values, and use the following additional meta-notational device: the expression *E0*{*E2*/*E1*} is well-formed iff *E0*, *E1* and *E2* are lvalues and *E1* occurs as a subexpression of *E0*. In that case, the expression denotes the result of substituting *E2* for all occurrences of *E1* in *E0*.

(II) $\dfrac{N2{:}P2 == G. \quad N1{:}P1 == L.}{N1{:}P1 == L\{G/N2{:}P2\}.}$

(III) $\dfrac{N2{:}P1 == G. \quad N1{:}P1 == L.}{N1{:}P1 == L\{G/N2\}.}$

(IV) $\dfrac{N1{:}P2 == G. \quad N1{:}P1 == L.}{N1{:}P1 == L\{G/P2\}.}$

Rule II says that if we have a theorem *N1*:*P1* == *L*. where *L* contains *N2*:*P2* as a subexpression, and we also have a theorem *N2*:*P2* == *G*., then we can derive a theorem in which all occurrences of *N2*:*P2* in *L* are replaced by *G*. In the simplest case, this means that we can interpret a sentence of the form

N1:*P1* == *N2*:*P2*.

as an inheritance specification meaning "the value of *P1* at *N1* is inherited from *P2* at *N2*". So for example, from:

NOUN: <sing gen> == s.
PRON: <sing gen> == **NOUN:**<sing gen>.

one can infer:

PRON: <sing gen> == s.

Rules III and IV are similar, but specify only a new node or path (not both) to inherit from. The other component (path or node) is unchanged, that is, it is the same as the corresponding component on the left-hand-side of the rule specifying the inheritance. In fact, the following two sentence schemas are entirely equivalent:

N1:*P1* == *N2*.
N1:*P1* == *N2*:*P1*.

as are these two:

N1:*P1* == *P2*.
N1:*P1* == *N1*:*P2*.

Rules II, III, and IV implement a local notion of inheritance in the sense that the new node or path specifications are interpreted in the current local context. The three remaining inference rules implement a non-local notion of inheritance: *quoted* descriptors specify values to be

interpreted in the context in which the original query was made (the *global* context), rather than the current context.

(V) $\dfrac{N2{:}P2 = V. \quad N1{:}P1 == G.}{N1{:}P1 = G\{V/"N2{:}P2"\}.}$

(VI) $\dfrac{N2{:}P1 = V. \quad N1{:}P1 == G.}{N1{:}P1 = G\{V/"N2"\}.}$

(VII) $\dfrac{N1{:}P2 = V. \quad N1{:}P1 == G.}{N1{:}P1 = G\{V/"P2"\}.}$

To see how the operation of these rules differs from the earlier unquoted cases, consider the following theory:

CAT: <sing> == <plur>.

V: <sing> == **CAT**
<plur> == er.

A1: <sing> == **CAT**
<plur> == ern.

A2: <sing> == en
<plur> == **A1**.

The intention here is that the CAT node expresses the generalisation that by default plural is the same as singular, V and A1 inherit this, but A2, while inheriting its plural form from A1, has an exceptional singular form, overriding inheritance from CAT (via A1). Now from this theory we can derive all the following theorems concerning plural:

V: <plur> = er.
A1: <plur> = ern.
A2: <plur> = ern.

and the following theorem concerning singular:

A2: <sing> = en.

But we cannot derive a theorem for V:<sing>, for example. This is because V:<sing> inherits from CAT:<sing>, which inherits (locally) from CAT:<plur>, which is not defined. What we wanted was for CAT:<sing> to inherit from V:<plur>, that is, from the *global* initial context. To achieve this we change the CAT definition to be:

CAT: <sing> == "<plur>".

Now we find that we can still derive the same plural theorems, but now in addition we get all these theorems concerning singular:

V: <sing> = er.
A1: <sing> = ern.
A2: <sing> = en.

For example, the derivation for the first of these is as follows:

(1) **V:**	<sing> == **CAT**.	(given)
(2) **CAT:**	<sing> == "<plur>".	(given)
(3) **V:**	<sing> == "<plur>".	(III on 1 and 2)
(4) **V:**	<plur> == er.	(given)
(5) **V:**	<plur> = er.	(I on 4)
(6) **V:**	<sing> = er.	(VII on 3 and 5)

Finally, given a set of sentences $\mathcal{T}$, we define the rule-closure of $\mathcal{T}$, **rcl**($\mathcal{T}$) to be the closure of $\mathcal{T}$ under finite application of the above inference rules in the conventional fashion.

4 Default inference

In addition to the conventional inference defined above, **DATR** has a nonmonotonic notion of inference by default: each definitional sentence about some node/path combination implicitly determines additional sentences about all the extensions to the path at that node for which no more specific definitional sentence exists in the theory. Our overall approach follows Moore (1983, 1985), whose treatment of inferences from sets of beliefs can be viewed more generally as a technique for providing a semantics for a declarative notion of inference by default (cf. Touretzky 1986, p34; Evans 1987). We begin with some auxiliary definitions.

The expression $P\hat{}Q$, where P and Q are paths, denotes the path formed by concatenating components of P and Q. A path $P2$ is an **extension** of a path $P1$ iff there is a path Q such that $P2 = P1\hat{}Q$. $P2$ is a **strict** extension iff Q is non-empty. We also use the ^ operator to denote extension of all the paths in a **DATR** sentence, as in the following examples:

S: N:<a> == v.
S^<c d>: N:<a c d> == v.

S: N1:<a> == N2:<x y>.
S^<c d>: N1:<a c d> == N2:<x y c d>.

S: N1:<a> == "N2:<>".
S^<c d>: N1:<a c d> == "N2:<c d>".

Given a sentence S, we define the **root** of S to be the [node]:[path] expression appearing to the left of the equality ('==' or '=') in S (for example the root of '*N:P* == *V.*' is '*N:P*)'. The root does not correspond to any syntactic category defined above: it is simply a substring of the sentence.

Given a set of sentences in **DATR**, $\mathcal{T}$, a node *N* and a path *P*, we say *N:P* is **specified** in $\mathcal{T}$ iff $\mathcal{T}$ contains a definitional sentence S whose root is *N:P*.

Let *N1:P1*, *N1:P2* be such that *N1:P1* is specified in $\mathcal{T}$. We say *N1:P2* is **connected** to *N1:P1* (relative to $\mathcal{T}$) iff:

i) *P2* is an extension of *P1*, and

ii) there is no strict extension *P3* of *P1* of which *P2* is an extension such that *N1:P3* is specified in $\mathcal{T}$.

So *N1:P2* is connected to *N1:P1* if *P1* is the maximal subpath of *P2* that is specified (with *N1*) in $\mathcal{T}$.

Now given a set of sentences $\mathcal{T}$, define the path closure **pcl**($\mathcal{T}$) of $\mathcal{T}$ to be:

pcl($\mathcal{T}$) = {S:S is an extensional sentence in $\mathcal{T}$}
∪ {S^Q: S is a definitional sentence in $\mathcal{T}$, with root *N:P*, and *N:P*^*Q* is connected to *N:P*}

It is clear from these definitions that any *N:P* is connected to itself and thus that $\mathcal{T}$ is always a subset of **pcl**($\mathcal{T}$). The path closure contains all those theorems which can be *inferred by default* from $\mathcal{T}$.

To illustrate path closure, consider the following example theory:

VERB:
<past> == ed
<past participle> == en.

We can infer by default the following theorems for **VERB**:

VERB:
<past> == ed
<past tense> == ed
<past participle> == en
<past tense singular> == ed
<past participle plural> == en
<past tense singular third> == ed.

The situation is slightly more complicated with sentences that have paths on their right-hand sides. Such paths are *also* extended by the subpath used to extend the left-hand side. So the sentence:

A2:<sing> == "**A1**:<plur>".

might give rise (by default) to sentences such as:

A2:<sing fem nom> == "**A1**:<plur fem nom>".

Using default inference, the example theory we used to illustrate global inference can be phrased more succinctly:

CAT: <sing> == "<plur>".

V: <> == **CAT**
<plur> == er.

A1: <> == **CAT**
<plur> == ern.

A2: <sing> == en
<> == **A1**.

In this version, we state that anything not specifically mentioned for **V** is inherited (by default) from **CAT**, whereas before we had to list cases (only 'sing' in the example) explicitly. Similarly **A1** inherits by default from **CAT**, and **A2** from **A1**. The operation of path closure is **non-monotonic**: if we add more sentences to our original theory, some of our derived sentences may cease to be true.

The two forms of inference in **DATR** are combined by taking the path closure of a theory first, and then applying the inference rules to the result. In other words, given a theory $\mathcal{T}$, and a sentence S, S is provable from $\mathcal{T}$ iff $S \in$ **rcl**(**pcl**($\mathcal{T}$)).

Acknowledgements

Evans's work was supported by a grant from the SERC. Gazdar's work was supported by grants from the ESRC and SERC. We are grateful to our referees and to Jon Cunningham, Walter Daelemans, David Israel, Bill Keller, Tom Khabaza, Ewan Klein, Bob Moore, Fernando Pereira, Allan Ramsay and Chris Thornton for clarifying our thinking about aspects of **DATR**.

References

Brachman, R. (1985) "I lied about the trees", or defaults and definitions in knowledge representation. *AI Magazine* 6.3, 80-93.

Calder, J. (1989) Paradigmatic morphology. *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, UMIST, April 1989. Morristown, NJ: ACL.

Calder, J. & E. te Lindert (1987) The protolexicon: towards a high-level language for lexical description. In Ewan Klein & Johan van Benthem, eds. *Categories, Polymorphism and Unification* Edinburgh/Amsterdam: CCS/ILLI, 356-370.

Daelemans, W.M.P. (1987a) A tool for the automatic creation, extension and updating of lexical knowledge bases. *ACL Proceedings, Third European Conference*, 70-74

Daelemans, W.M.P. (1987b) *Studies in language technology: an object-oriented computer model of morphonological aspects of Dutch.* Doctoral dissertation, Catholic University of Leuven.

de Smedt, K. (1984) Using object-oriented knowledge-representation techniques in morphology and syntax programming. In T. O'Shea (ed.) *ECAI-84: Proceedings of the Sixth European Conference on Artificial Intelligence* Amsterdam: Elsevier, 181-184.

Etherington, D.W. (1988) *Reasoning with Incomplete Information.* Los Altos: Morgan Kaufmann.

Evans, R. (1987) Towards a formal specification for defaults in GPSG. In Ewan Klein & Johan van Benthem, eds. *Categories, Polymorphism and Unification.* Edinburgh/Amsterdam: CCS/ILLI, 73-93.

Evans, R. & Gazdar, G. (1989) The semantics of **DATR**. In A. Cohn (ed.) *AISB-89, Proceeedings of the Seventh Conference of the Society for the Study of Artificial Intelligence and Simulation of Behaviour.* London: Pitman.

Flickinger, D., Pollard, C.J. & Wasow, T. (1985) Structure sharing in lexical representation. *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics* (Chicago), 262-267.

Gazdar, G. (1987) Linguistic applications of default inheritance mechanisms. In Peter J. Whitelock et al., eds. *Linguistic Theory and Computer Applications.* London: Academic Press, 37-67.

Moore, R.C. (1983) Semantical considerations on nonmonotonic logic. *Technical Note 284*, SRI International, Menlo Park. Revised and expanded version of a paper that appeared in *IJCAI-83*, 272-279.

Moore, R.C. (1985) Possible-worlds semantics for autoepistemic logic. *Report No. CSLI-85-41*, Center for the Study of Language and Information, Stanford. Also published in the *Proceedings of the AAAI Non-Monotonic Reasoning Workshop*, 344-354.

Shieber, S.M. (1986) *An Introduction to Unification Approaches to Grammar.* Stanford: CSLI/Chicago University Press.

Touretzky, D.F. (1986) *The Mathematics of Inheritance Systems.* Los Altos: Morgan Kaufmann.