DATR Theories and DATR Models

Bill Keller
School of Cognitive and Computing Sciences
The University of Sussex
Brighton, UK
email: billk@cogs.susx.ac.uk

Abstract

Evans and Gazdar (Evans and Gazdar, 1989a; Evans and Gazdar, 1989b) introduced DATR as a simple, non-monotonic language for representing natural language lexicons. Although a number of implementations of DATR exist, the full language has until now lacked an explicit, declarative semantics. This paper rectifies the situation by providing a mathematical semantics for DATR. We present a view of DATR as a language for defining certain kinds of partial functions by cases. The formal model provides a transparent treatment of DATR's notion of global context. It is shown that DATR's default mechanism can be accounted for by interpreting value descriptors as families of values indexed by paths.

1 Introduction

DATR was introduced by Evans and Gazdar (1989a; 1989b) as a simple, declarative language for representing lexical knowledge in terms of path/value equations. The language lacks many of the constructs found in general purpose, knowledge representation formalisms, yet it has sufficient expressive power to capture concisely the structure of lexical information at a variety of levels of linguistic description. At the present time, DATR is probably the most widely-used formalism for representing natural language lexicons in the natural language processing (NLP) community. There are around a dozen different implementations of the language and large DATR lexicons have been constructed for use in a variety of applications (Cahill and Evans, 1990; Andry et al., 1992; Cahill, 1994). DATR has been applied to problems in inflectional and derivational morphology (Gazdar, 1992; Kilbury, 1992; Corbett and Fraser, 1993), lexical semantics (Kilgariff, 1993), morphonology (Cahill, 1993), prosody (Gibbon and Bleiching, 1991) and speech (Andry et al., 1992). In more recent work, the language has been used to provide a concise encoding of Lexicalised Tree Adjoining Grammar (Evans et al., 1994; Evans et al., 1995).

A primary objective in the development of DATR has been the provision of an explicit, mathematically rigorous semantics. This goal was addressed in one of the first publications on the language (Evans and Gazdar, 1989b). The definitions given there deal with a subset of DATR that includes core features of the language such as the notions of local and global inheritance and DATR's default mechanism. However, they exclude some important and widely-used constructs, most notably string (or 'list') values and evaluable paths. Moreover, it is by no means clear that the approach can be generalized appropriately to cover these features. In particular, the formal apparatus introduced by Evans and Gazdar in (1989b) provides no explicit model of DATR 's notion of *global context*. Rather, local and global inheritance are represented by distinct semantic functions $\mathcal{L}$ and $\mathcal{G}$. This approach is possible only on the (overly restrictive) assumption that DATR statements involve either local or global inheritance relations, but never both.

The purpose of the present paper is to remedy the deficiencies of the work described in (Evans and Gazdar, 1989b) by furnishing DATR with a transparent, mathematical semantics. There is a standard view of DATR as a language for representing a certain class of non-monotonic inheritance networks ('semantic nets'). While this perspective provides an intuitive and appealing way of thinking about the structure and representation of lexical knowledge, it is less clear that it provides an accurate or particularly helpful picture of the DATR language itself. In fact, there are a number of constructs available in DATR that are impossible to visualize in terms of simple inheritance hierarchies. For this reason, the work described in this paper reflects a rather different perspective on DATR, as a language for defining certain kinds of partial functions by cases. In the following sections this viewpoint is made more precise. Section 2 presents the syntax of the DATR language and introduces the notion of a DATR theory. An informal introduction to the DATR language is provided, by example, in section 3. The semantics of DATR is then covered in two stages. Section 4.1 introduces DATR interepretations and describes the semantics of a restricted version of the language without defaults. The treatment of implicit information is covered in section 4.2, which provides a definition of a default model for a DATR theory.

2 DATR Theories

Let NODE and ATOM be disjoint sets of symbols (the *nodes* and *atoms* respectively). Nodes are denoted by N and atoms by a. The set DESC of DATR *value descriptors* (or simply *descriptors*) is built up from the atoms and nodes as shown below. Descriptors are denoted by d.

- $a \in \text{DESC}$ for any $a \in \text{ATOM}$
- For any $N \in \text{NODE}$ and $d_1 \dots d_n \in \text{DESC}$:

$$N : \langle d_1 \cdots d_n \rangle \in \text{DESC}$$
$$\text{“}N : \langle d_1 \cdots d_n \rangle\text{”} \in \text{DESC}$$
$$\text{“}\langle d_1 \cdots d_n \rangle\text{”} \in \text{DESC}$$
$$\text{“}N\text{”} \in \text{DESC}$$

Value descriptors are either atoms or *inheritance descriptors*, where an inheritance descriptor is further distinguished as either *local* (unquoted) or *global* (quoted). There is just one kind of local descriptor (node/path), but three kinds of global descriptor (node/path, path and node) [1].

A *path* $\langle a_1 \dots a_n \rangle$ is a (possibly empty) sequence of atoms enclosed in angle brackets. Paths are denoted by P. For N a node, P a path and $\alpha \in \text{ATOM}^*$ a (possibly empty) sequence of atoms, an equation of the form $N : P = \alpha$ is called an *extensional sentence*. Intuitively, an extensional sentence $N : P = \alpha$ states that the value associated with the path P at node N is α. For ϕ a (possibly empty) sequence of value descriptors, an equation of the form $N : P == \phi$ is called a *definitional sentence*. A definitional sentence $N : P == \phi$ specifies a property of the node N, namely that the path P is associated with the value defined by the sequence of value descriptors ϕ.

A collection of equations can be used to specify the properties of different nodes in terms of one another, and a finite set of DATR sentences $\mathcal{T}$ is called a DATR *theory*. In principle, a DATR theory $\mathcal{T}$ may consist of any combination of DATR sentences, either definitional or extensional, but in practice, DATR theories are more restricted than this. The theory $\mathcal{T}$ is said to be *definitional* if it consists solely of definitional sentences and it is said to be *functional* if it meets the following condition:

[1]The syntax presented in (Evans and Gazdar, 1989a; Evans and Gazdar, 1989b) permits nodes and paths to stand as local descriptors. However, these additional forms can be viewed as conventional abbreviations, in the appropriate syntactic context, for node/path pairs

$$N : P == \phi \text{ and } N : P == \psi \in \mathcal{T} \text{ implies } \phi = \psi$$

There is a pragmatic distinction between definitional and extensional sentences akin to that drawn between the language used to define a database and that used to query it. DATR interpreters conventionally treat all extensional sentences as 'goal' statements, and evaluate them as soon as they are encountered. Thus, it is not possible, in practice, to combine definitional and extensional sentences within a theory[2]. Functionality for DATR theories, as defined above, is really a syntactic notion. However, it approximates a deeper, semantic requirement that the nodes should correspond to (partial) functions from paths to values.

In the remainder of this paper we will use the term (DATR) *theory* always in the sense *functional, definitional* (DATR) *theory*. For a given DATR theory $\mathcal{T}$ and node N of $\mathcal{T}$, we write $\mathcal{T}/N$ to denote that subset of the sentences in $\mathcal{T}$ that relate to the node N. That is:

$$\mathcal{T}/N \quad = \quad \{s \in \mathcal{T} | s = N : P == \phi\}$$

The set $\mathcal{T}/N$ is referred to as the *definition of* N *(in* $\mathcal{T}$*)*.

3 An Overview of DATR

An example of (a fragment of) a DATR theory is shown in figure 1. The theory makes use of some standard abbreviatory devices that enable nodes and/or paths to be omitted in certain cases. For example, sets of sentences relating to the same node are written with the node name implicit in all but the first-given sentence in the set. Also, we write **See** : $\langle\rangle$ == **Verb** to abbreviate the definitional sentence **See** : $\langle\rangle$ == **Verb** : $\langle\rangle$, and similarly elsewhere.

The theory defines the properties of seven nodes: an abstract **Verb** node, nodes **EnVerb**, **Aux** and **Modal**, and three abstract lexemes **Walk**, **Mow** and **Can**. Each node is associated with a collection of definitional sentences that specify values associated with different paths. This specification is achieved either *explicitly*, or *implicitly*. Values given explicitly are specified either *directly*, by exhibiting a particular value, or *indirectly*, in terms of local and/or global inheritance. Implicit specification is achieved via DATR 's default mechanism.

For example, the definition of the **Verb** node gives the values of the paths ⟨**syn cat**⟩ and ⟨**syn type**⟩ directly, as **verb** and **main**, respectively. Similarly, the definition of **Walk** gives the value of ⟨**mor root**⟩ directly as **walk**. On the other hand, the value of

[2]It is not clear why one would wish to do this anyway, but the possibility is explicitly left open in the original definitions of (Evans and Gazdar, 1989a).

```
Verb :      ⟨syn cat⟩ == verb
            ⟨syn type⟩ == main
            ⟨mor form⟩ == "⟨mor "⟨syn form⟩"⟩"
            ⟨mor pres⟩ == "⟨mor root⟩"
            ⟨mor past⟩ == "⟨mor root⟩" ed
            ⟨mor pres part⟩ == "⟨mor root⟩" ing
            ⟨mor pres sing three⟩ == "⟨mor root⟩" s

EnVerb :    ⟨⟩ == Verb
            ⟨mor past part⟩ == "⟨mor root⟩" en

Aux :       ⟨⟩ == Verb
            ⟨syn type⟩ == aux

Modal :     ⟨⟩ == Aux
            ⟨mor pres sing three⟩ == "⟨mor root⟩"

Walk :      ⟨⟩ == Verb
            ⟨mor root⟩ == walk

Mow :       ⟨⟩ == EnVerb
            ⟨mor root⟩ == mow

Can :       ⟨⟩ == Modal
            ⟨mor root⟩ == can
            ⟨mor past⟩ == could
```

Figure 1: A DATR Theory

the empty path at **Walk** is given indirectly, by local inheritance, as the value of the empty path at **Verb**. Note that in itself, this might not appear to be particularly useful, since the theory does not provide an explicit value for the empty path in the definition of **Verb**. However, DATR's default mechanism permits any definitional sentence to be applicable not only to the path specified in its left-hand-side, but also for any rightward extension of that path for which no more specific definitional sentences exist. This means that the statement **Walk** : ⟨⟩ == **Verb** : ⟨⟩ actually corresponds to a class of *implicit* definitional sentences, each obtained by extending paths on the left- and the right-hand-sides of the equation in the same manner. Examples include the following:

Walk : ⟨**mor**⟩ == **Verb** : ⟨**mor**⟩
Walk : ⟨**mor form**⟩ == **Verb** : ⟨**mor form**⟩
Walk : ⟨**syn cat**⟩ == **Verb** : ⟨**syn cat**⟩

Thus, the value associated with ⟨**syn cat**⟩ at **Walk** is given (implicitly) as the value of ⟨**syn cat**⟩ at **Verb**, which is given (explicitly) as **verb**. Also, the values of ⟨**mor**⟩ and ⟨**mor form**⟩, amongst many others, are inherited from **Verb**. In the same way, the value of ⟨**syn cat**⟩ at **Mow** is inherited locally from **EnVerb** (which in turn inherits locally from **Verb**) and the value of ⟨**syn cat**⟩ at **Can** is inherited locally from **Modal** (which ultimately gets its value from **Verb** via **Aux**). Note however, that the following sentences do *not* follow by default from the specifications given at the relevant nodes:

Walk : ⟨**mor root**⟩ == **Verb** : ⟨**mor root**⟩
Can : ⟨**mor past**⟩ == **Modal** : ⟨**mor past**⟩
Aux : ⟨**syn type**⟩ == **Verb** : ⟨**syn type**⟩

In each of the above cases, the theory provides an explicit statement about the value associated with the indicated path at the given node. As a result the default mechanism is effectively over-ridden.

In order to understand the use of global (i.e. quoted) inheritance descriptors it is necessary to introduce DATR's notion of a *global context*. Suppose then that we wish to determine the value associated with the path ⟨**mor pres**⟩ at the node **Walk**. In this case, the global context will initially consist of the node/path pair **Walk**/⟨**mor pres**⟩. Now, by default the value associated with ⟨**mor pres**⟩ at **Walk** is inherited locally from ⟨**mor pres**⟩ at **Verb**. This, in turn, inherits *globally* from the path ⟨**mor root**⟩. That is:

Verb : ⟨**mor pres**⟩ == "⟨**mor root**⟩"

Consequently, the required value is that associated with ⟨**mor root**⟩ at the 'global node' **Walk** (i.e. the node provided by the current global context), which is just **walk**. In a similar fashion, the value

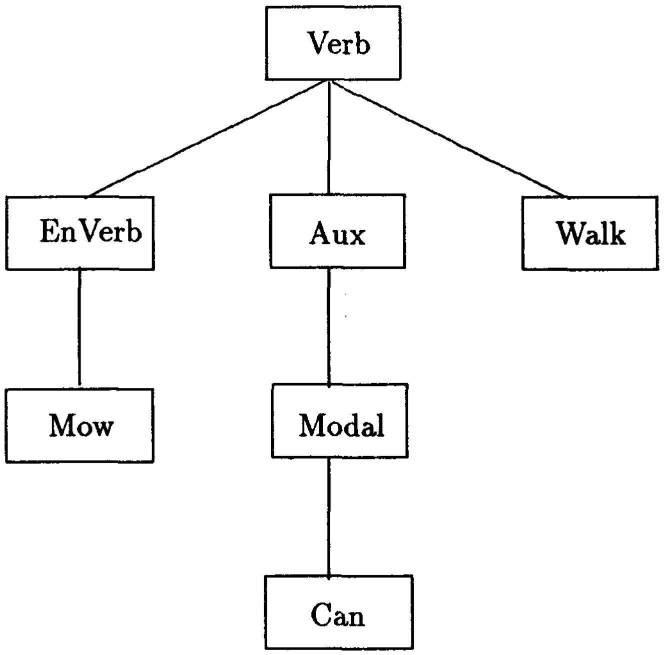

Figure 2: A Lexical Inheritance Hierarchy

associated with ⟨**mor past**⟩ at **Walk** is obtained as **walk ed** (i.e. the string of atoms formed by evaluating the specification "⟨**mor root**⟩" ed in the global context **Walk**/⟨**mor past**⟩).

More generally, the global context is used to fill in the missing node (path) when a global path (node) is encountered. In addition however, the evaluation of a global descriptor results in the global context being set to the new node/path pair. Thus in the preceding example, after the quoted descriptor "⟨**mor root**⟩" is encountered, the global context effectively becomes **Walk** / ⟨**mor root**⟩ (i.e. the path component of the global context is altered). Note that there is a real distinction between a local inheritance descriptor of the form $N : P$ and it's global counterpart "$N : P$". The former has no effect on the global context, while the latter effectively overwrites it.

Finally, the definition of **Verb** in the theory of figure 1 illustrates a use of the 'evaluable path' construct:

Verb : ⟨**mor form**⟩ == "⟨**mor** "⟨**syn form**⟩"⟩"

This states that the value of ⟨**mor form**⟩ at **Verb** is inherited globally from the path ⟨**mor** · · ·⟩, where the dots represent the result of evaluating the global path "⟨**syn form**⟩" (i.e. the value associated with ⟨**syn form**⟩ in the prevailing global context). Evaluable paths provide a powerful means of capturing generalizations about the structure of lexical information.

4 DATR Models

To a first level of approximation, the DATR theory of figure 1 can be understood as a representation of an inheritance hierarchy (a 'semantic network') as shown in figure 2. In the diagram, nodes are written as labelled boxes, and arcs correspond to (local) inheritance, or *isa* links. Thus, the node **Can** inherits from **Modal** which inherits from **Aux** which in turn is a **Verb**. The hierarchy provides a useful means of visualising the overall structure of the lexical knowledge encoded by the DATR theory. However, the semantic network metaphor is of far less value as a way of thinking about the DATR language itself. Note that there is nothing inherent in DATR to ensure that theories correspond to simple *isa* hierarchies of the kind shown in the figure. What is more, the DATR language includes constructs that cannot be visualized in terms of simple networks of nodes connected by (local) inheritance links. Global inheritance, for example, has a dynamic aspect which is difficult to represent in terms of static links. Similar problems are presented by both string values and evaluable paths. Our conclusion is that the network metaphor is of primary value to the DATR user. In order to provide a satisfactory, formal model of how the language 'works' it is necessary to adopt a different perspective.

DATR theories can be viewed semantically as collections of definitions of partial functions ('nodes' in DATR parlance) that map paths onto values. A model of a DATR theory is then an assignment of functions to node symbols that is consistent with the definitions of those nodes within the theory. This picture of DATR as a formalism for defining partial functions is complicated by two features of the language however. First, the meaning of a given node depends, in general, on the global context of interpretation, so that nodes do not correspond directly to mappings from paths to values, but rather to functions from *contexts* to such mappings. Second, it is necessary to provide an account of DATR's default mechanism. It will be convenient to present our account of the semantics of DATR in two stages.

4.1 DATR Interpretations

This section considers a restricted version of DATR without the default mechanism. Section 4.2 then shows how implicit information can be modelled by treating value descriptors as families of values indexed by paths.

Definition 4.1 *A* DATR interpretation *is a triple* $I = (U, \kappa, F)$, *where*

1. *U is a set;*
2. *κ is a function assigning to each element of the set $(U \times U^*)$ a partial function from $(U \times U^*)$ to U^*.*
3. *F is a valuation function assigning to each node N and atom a an element of U, such that distinct atoms are assigned distinct elements.*

Elements of the set U are denoted by u and elements of U^* are denoted by v. Intuitively, U^* is the domain of (semantic) values/paths. Elements of the set $C = (U \times U^*)$ are called *contexts* and denoted by c. The function κ can be thought of as mapping global contexts onto (partial) functions from local contexts to values. The function F is extended to paths, so that for $P = \langle a_1 \cdots a_n \rangle$ $(n \geq 0)$ we write $F(P)$ to denote $u_1 \cdots u_n \in U^*$, where $u_i = F(a_i)$ for each i $(1 \leq i \leq n)$.

Intuitively, value descriptors denote elements of U^* (as we shall see, this will need to be revised later in order to account for DATR's default mechanism). We associate with the interpretation $I = (U, \kappa, F)$ a partial denotation function $[\![]\!] : \text{DESC} \rightarrow (C \rightarrow U^*)$ and write $[\![d]\!]_c$ to denote the meaning (value) of descriptor d in the global context c. The denotation function is defined as shown in figure 3. Note that an atom always denotes the same element of U, regardless of the context. By contrast, the denotation of an inheritance descriptor is, in general, sensitive to the global context c in which it appears. Note also that in the case of a global inheritance descriptor, the global context is effectively altered to reflect the new local context c'. The denotation function is extended to sequences of value descriptors in the obvious way. Thus, for $\phi = d_1 \cdots d_n$ $(n \geq 0)$, we write $[\![\phi]\!]_c$ to denote $v_1 \cdots v_n \in U^*$ if $v_i = [\![d_i]\!]_c$ $(1 \leq i \leq n)$ is defined (and $[\![\phi]\!]_c$ is undefined otherwise).

Now, let $I = (U, \kappa, F)$ be an interpretation and $\mathcal{T}$ a theory. We will write $[\mathcal{T}/N]_c$ to denote that partial function from U^* to U^* given by

$$[\mathcal{T}/N]_c = \bigcup_{N:P==\phi \in \mathcal{T}} \{(F(P), [\phi]_c)\}$$

It is easy to verify that $[\mathcal{T}/N]_c$ does indeed denote a partial function (it follows from the functionality of the theory $\mathcal{T}$). Let us also write $[N]_c$ to denote that partial function from U^* to U^* given by $[N]_c(v) = \kappa(c)(F(N), v)$, for all $v \in U^*$. Then, I *models* $\mathcal{T}$ just in case the following containment holds for each node N and context c:

$$[N]_c \supseteq [\mathcal{T}/N]_c$$

That is, an interpretation is a model of a DATR theory just in case (for each global context) the function it associates with each node respects the definition of that node within the theory.

4.2 Implicit Information and Default Models

The notion of a model presented in the preceding section is too liberal in that it takes no account of information *implicit* in a theory. For example, consider again the definition of the node **Walk** from the theory of figure 1, and repeated below.

$$\begin{array}{ll} \textbf{Walk}: & \langle\rangle == \textbf{Verb} \\ & \langle \textbf{mor root} \rangle == \textbf{walk} \end{array}$$

According to the definition of a model given previously, any model of the theory of figure 1 will associate with the node **Walk** a function from paths to values which respects the above definition. This means that for every global context c, the following containment must hold[3]:

$$\begin{array}{rcl} [\textbf{Walk}]_c & \supseteq & \{\langle\langle\rangle, [\textbf{Verb} : \langle\rangle]_c\rangle, \\ & & \langle\langle \textbf{mor root}\rangle, \textbf{walk}\rangle\} \end{array}$$

On the other hand, there is no guarantee that a given model will also respect the following containment:

$$\begin{array}{rcl} [\textbf{Walk}]_c & \supseteq & \{\langle\langle \textbf{mor}\rangle, [\textbf{Verb} : \langle \textbf{mor}\rangle]_c\rangle, \\ & & \langle\langle \textbf{mor root root}\rangle, \textbf{walk}\rangle\} \end{array}$$

In fact, this containment (amongst other things) *should* hold. It follows 'by default' from the statements made about **Walk** that the path $\langle$**mor**$\rangle$ inherits locally from **Verb** and that the value associated with any extension of $\langle$**mor root**$\rangle$ is **walk**.

[3]In this and subsequent examples, syntactic objects (e.g.**walk**, $\langle$**mor root**$\rangle$) are used to stand for their semantic counterparts under F (i.e. F(**walk**), F($\langle$**mor root**$\rangle$), respectively).

$$\begin{array}{lcl}
[\![a]\!]_c & = & F(a) \\
[\![N : \langle d_1 \cdots d_n \rangle]\!]_c & = & \left\{ \begin{array}{l} \text{if } v_i = [\![d_i]\!]_c \text{ is defined for each } i\,(1 \leq i \leq n), \text{ then} \\ \kappa(c)(F(N), v_1 \cdots v_n) \\ \text{undefined otherwise} \end{array} \right. \\
[\![\text{“}N : \langle d_1 \cdots d_n \rangle\text{”}]\!]_c & = & \left\{ \begin{array}{l} \text{if } v_i = [\![d_i]\!]_c \text{ is defined for each } i\,(1 \leq i \leq n), \text{ then} \\ \kappa(c')(c') \text{ where } c' = (F(N), v_1 \cdots v_n) \\ \text{undefined otherwise} \end{array} \right. \\
[\![\text{“}\langle d_1 \cdots d_n \rangle\text{”}]\!]_c & = & \left\{ \begin{array}{l} \text{if } v_i = [\![d_i]\!]_c \text{ is defined for each } i\,(1 \leq i \leq n), \text{ then} \\ \kappa(c')(c') \text{ where } c = (u, v) \text{ and } c' = (u, v_1 \cdots v_n) \\ \text{undefined otherwise} \end{array} \right. \\
[\![\text{“}N\text{”}]\!]_c & = & \kappa(c')(c') \text{ where } c = (u, v) \text{ and } c' = (F(N), v)
\end{array}$$

Figure 3: Denotation function for DATR Descriptors

There have been a number of formal treatments of defaults in the setting of attribute-value formalisms (Carpenter, 1993; Bouma, 1992; Russell et al., 1992; Young and Rounds, 1993). Each of these approaches formalizes a notion of default inheritance by defining appropriate operations (e.g. default unification) for combining strict and default information. Strict information is allowed to over-ride default information where the combination would otherwise lead to inconsistency (i.e. unification failure). In the case of DATR however, the formalism does not draw an explicit distinction between strict and default values for paths. In fact, all of the information given explicitly in a DATR theory is strict. The non-monotonic nature of DATR theories arises from a general, default mechanism which 'fills in the gaps' by supplying values for paths not explicitly specified in a theory. More specifically, DATR's default mechanism ensures that any path that is not explicitly specified for a given node will take its definition from the longest prefix of that path that *is* specified. Thus, the default mechanism defines a class of implicit, definitional sentences with paths on the left that extend paths found on the left of explicit sentences. Furthermore, this extension of paths is also carried over to paths occurring on the right. In effect, each (explicit) path is associated not just with a single value specification, but with a whole family of specifications indexed by extensions of those paths.

This suggests the following approach to the semantics of defaults in DATR. Rather than interpreting node definitions (in a given global context) as partial functions from paths to values (i.e. of type $U^* \rightarrow U^*$) we choose instead to interpret them as partial functions from (explicit) paths, to functions from extensions of those paths to values (i.e. of type $U^* \rightarrow (U^* \rightarrow U^*)$). Now suppose that $f : U^* \rightarrow (U^* \rightarrow U^*)$ is the function associated with the node definition $\mathcal{T}/N$ in a given DATR interpretation. We can define a partial function $\Delta(f) : U^* \rightarrow U^*$ (the ***default interpretation*** of $\mathcal{T}/N$) as follows. For each $v \in U^*$ set

$$\Delta(f)(v) \quad = \quad f(v_1)(v_2)$$

where $v = v_1 v_2$ and v_1 is the longest prefix of v such that $f(v_1)$ is defined. In effect, the function $\Delta(f)$ makes explicit that information about paths and values that is only implicit in f, but just in so far as it does not conflict with explicit information provided by f.

In order to re-interpret node definitions in the manner suggested above, it is necessary to modify the interpretation of value descriptors. In a given global context c, a value descriptor d now corresponds to a total function $[\![d]\!]_c : U^* \rightarrow U^*$ (intuitively, a function from path extensions to values). For example, atoms now denote constant functions:

$$[\![a]\!]_c(v) \quad = \quad F(a) \text{ for all } v \in U^*$$

More generally, value descriptors will denote different values for different paths. Figure 4 shows the revised clause for global node/path pairs, the other definitions being very similar. Note the way in which the 'path' argument v is used to extend $v_1 \cdots v_n$ in order to define the new local (and in this case also, global) context c'. On the other hand, the meaning of each of the d_i is obtained with respect to the 'empty path' ϵ (i.e. path extension does not apply to subterms of inheritance descriptors).

As before, the interpretation function is extended to sequences of path descriptors, so that for $\phi = d_1 \cdots d_n$ $(n \geq o)$ we have $[\![\phi]\!]_c(v) = v_1 \cdots v_n \in U^*$, if $v_i = [\![d_i]\!](v)$ is defined, for each i $(1 \leq i \leq n)$ (and $[\![\phi]\!]_c(v)$ is undefined otherwise). The definition of the interpretation of node definitions can be taken over unchanged from the previous section. However, for a theory $\mathcal{T}$ and node N, the function $[\![\mathcal{T}/N]\!]_c$ is now of type $U^* \rightarrow (U^* \rightarrow U^*)$. An interpretation $I = (U, \kappa, F)$ is a ***default model*** for theory $\mathcal{T}$ just in case for every context c and node N we have:

$$[\![N]\!]_c \quad \supseteq \quad \Delta([\![\mathcal{T}/N]\!]_c)$$

As an example, consider the default interpretation of the definition of the node **Walk** given above. By

$$[\![\text{“}N : \langle d_1 \cdots d_n \rangle\text{”}]\!]_c(v) = \begin{cases} \text{if } v_i = [\![d_i]\!]_c(\epsilon) \text{ is defined for each } i (1 \leq i \leq n), \text{ then} \\ \kappa(c')(c') \text{ where } c' = (F(N), v_1 \cdots v_n v) \\ \text{undefined otherwise} \end{cases}$$

Figure 4: Revised denotation for global node/path pairs

definition, any default model of the theory of figure 1 must respect the following containment:

$$[\![\mathbf{Walk}]\!]_c \supseteq \Delta(\{\langle\langle\rangle, \lambda v.[\![\mathbf{Verb} : \langle\rangle]\!]_c(v)\rangle, \langle\langle \mathbf{mor\ root}\rangle, \lambda v.\mathbf{walk}\rangle\}$$

¿From the definition of Δ, it follows that for any path v, if v extends $\langle$mor root$\rangle$, then it is mapped onto the value walk, and otherwise it is mapped to the value given by $[\![\text{Verb} : \langle\rangle]\!]_c(v)$. We have the following picture:

$$\begin{aligned}[\![\mathbf{Walk}]\!]_c \supseteq \{ & \langle\langle\rangle, [\![\mathbf{Verb} : \langle\rangle]\!]_c(\langle\rangle)\rangle, \\ & \langle\langle \mathbf{mor}\rangle, [\![\mathbf{Verb} : \langle\rangle]\!]_c(\langle \mathbf{mor}\rangle)\rangle, \\ & \ldots \\ & \langle\langle \mathbf{mor\ root}\rangle, \mathbf{walk}\rangle, \\ & \langle\langle \mathbf{mor\ root\ root}\rangle, \mathbf{walk}\rangle, \\ & \ldots\}\end{aligned}$$

The default models of a theory $\mathcal{T}$ constitute a proper subset of the models of $\mathcal{T}$: just those that respect the default interpretations of each of the nodes defined within the theory.

5 Conclusions

The work described in this paper fulfils one of the objectives of the DATR programme: to provide the language with an explicit, declarative semantics. We have presented a formal model of DATR as a language for defining partial functions and this model has been contrasted with an informal view of DATR as a language for representing inheritance hierarchies. The approach provides a transparent treatment of DATR's notion of (local and global) context and accounts for DATR's default mechanism by regarding value descriptors (semantically) as families of values indexed by paths.

The provision of a formal semantics for DATR is important for several reasons. First, it provides the DATR user with a concise, implementation-independent account of the meaning of DATR theories. Second, it serves as a standard against which other, operational definitions of the formalism can be judged. Indeed, in the absence of such a standard, it is impossible to demonstrate formally the correctness of novel implementation strategies (for an example of such a strategy, see (Langer, 1994)). Third, the process of formalisation itself aids our understanding of the language and its relationship to other non-monotonic, attribute-value formalisms. Finally, the semantics presented in this paper provides a sound basis for subsequent investigations into the mathematical and computational properties of DATR.

6 Acknowledgements

The author would like to thank Roger Evans, Gerald Gazdar, Bill Rounds and David Weir for helpful discussions on the work described in this paper.

References

Francois Andry, Norman Fraser, Scott McGlashan, Simon Thornton, and Nick Youd. 1992. Making DATR work for speech: lexicon compilation in SUNDIAL. *Computational Linguistics*, 18(3):245–267.

Gosse Bouma. 1992. Feature structures and nonmonotonicity. *Computational Linguistics*, 18(2):183–203.

Lynne Cahill and Roger Evans. 1990. An application of DATR: the TIC lexicon. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 120–125.

Lynne Cahill. 1993. Morphonology in the lexicon. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*, pages 87–96.

Lynne Cahill. 1994. An inheritance-based lexicon for message understanding systems. In *Proceedings of the 4th ACL Conference on Applied Natural Language Processing*, pages 211–212.

Bob Carpenter. 1993. Skeptical and credulous default unification with applications to templates and inheritance. In Ted Briscoe, Valeria de Paiva, and Ann Copestake, editors, *Inheritance, Defaults and the Lexicon*, pages 13–37. Cambridge University Press, Cambridge.

Greville Corbett and Norman Fraser. 1993. Network morphology: a DATR account of Russian nominal inflection. *Journal of Linguistics*, 29:113–142.

Roger Evans and Gerald Gazdar. 1989a. Inference in DATR. In *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics*, pages 66–71.

Roger Evans and Gerald Gazdar. 1989b. The semantics of DATR. In *Proceedings of AISB-89*, pages 79–87.

Roger Evans, Gerald Gazdar, and David Weir. 1994. Using default inheritance to describe LTAG. In *3e Colloque International sur les Grammaires d'Arbres Adjoints (TAG+3)*, pages 79–87.

Roger Evans, Gerald Gazdar, and David Weir. 1995. Encoding lexicalized tree adjoining grammars with a nonmonotonic inheritance hierarchy. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics.*

Gerald Gazdar. 1992. Paradigm function morphology in DATR. In Lynne Cahill and Richard Coates, editors, *Sussex Papers in General and Computational Linguistics*, number CSRP 239 in Cognitive Science Research Papers, pages 45–53. University of Sussex, Brighton.

Dafydd Gibbon and Doris Bleiching. 1991. An ILEX model for German compound stress in DATR. In *Proceedings of the FORWISS-ASL Workshop on Prosody in Man-Machine Communication.*

James Kilbury. 1992. Pardigm-based derivational morphology. In Guenther Goerz, editor, *Proceedings of KONVENS 92*, pages 159–168. Springer, Berlin.

Adam Kilgariff. 1993. Inheriting verb alternations. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*, pages 213–221.

Hagen Langer. 1994. Reverse queries in DATR. In *Proceedings of the 15th International Conference on Computational Linguistics*, volume II, pages 1089–1095, Kyoto.

Graham Russell, Afzal Ballim, John Carroll, and Susan Warwick-Armstrong. 1992. A practical approach to multiple default inheritance for unification-based lexicons. *Computational Linguistics*, 18(2):311–337.

Mark Young and Bill Rounds. 1993. A logical semantics for nonmonotonic sorts. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 209–215.