BMBF

Verbmobil
Verbundvorhaben

DATR without nodes and global inheritance

Hagen Langer

Universität Bielefeld

Vm

Report 14
Juni 1994

Juni 1994

Hagen Langer

Universität Bielefeld (UBI)
Universitätsstr. 25
Postfach 10 01 31
33501 Bielefeld

Universität Osnabrück
FB 7: Sprach- und Literaturwissenschaft
Neuer Graben 40
49069 Osnabrück

Tel.: (0521) 106 - 3510
Fax: (0521) 106 - 6008
e-mail: hlanger@jupiter.rz.uni-osnabrueck.de

Gehört zum Antragsabschnitt: 5

Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) unter dem Förderkennzeichen 01 IV 101 B 2 gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei dem Autor.

Abstract

In this paper we will investigate which elements of the DATR language essentially contribute to its expressive capabilities and which are dispensable for the purposes DATR has been developed for. We will consider a subset of DATR, called *local path DATR (LDATR)*, that eliminates the concepts of *node* and *global inheritance* by redefining them in a pseudo-bootstrapping manner in terms of the local path inheritance alone. We will argue that, for an arbitrary standard DATR theory θ_1, there is a LDATR theory θ_2 such that each theorem of θ_1 corresponds to an equivalent theorem of θ_2. We show this by general translation rules which map an arbitrary standard DATR theory onto its LDATR counterpart. The main result of the paper is that restricting DATR to the rules of inference I and IV, yields a still DATR equivalent, and hence, still Turing equivalent formalism. Furthermore, we argue that a version of LDATR with variables is strongly equivalent to a substantial subset of standard DATR. Finally, some consequences of using the LDATR approach as a modelling convention for lexicon development in DATR are discussed[1].

---

[1] We would like to thank Roger Evans, Gerald Gazdar, and Dafydd Gibbon for very useful comments on an earlier draft of this paper. Thanks are also due to John Nerbonne, and the anonymous reviewers of the DGfS/CL Conference 'Deklarative und prozedurale Aspekte der Sprachverarbeitung', Hamburg, 17th-19th November 1993, where the basic ideas of this report were presented as an oral talk. Many aspects of the present paper were inspired by previous work of Lionel Moser, Petra Naerger, and Ingrid Renz

Contents

Chapter 1

Introduction

DATR (Evans & Gazdar 1989) has become one of the most widely used formal languages for the representation of lexical information. DATR applications have been developed for a wide variety of languages (including English, Japanese, Kikuyu, Arabic, Latin, and others) and many different subdomains of lexical representation, including

- inflectional morphology,
- underspecification phonology,
- non-concatenative morphophonology,
- lexical semantics, and
- tone systems[1].

We presuppose that the reader of the present paper is familiar with the basic features of DATR as specified in Evans & Gazdar [1989a]. One of the claims which have been stated for DATR is that it provides very compact lexical representations which eliminate undesirable redundancies. Therefore it seems to be in the spirit of the DATR enterprise to investigate if DATR itself, as a formal language, exhibits redundancies, or not. In the present paper we will argue that there are in fact considerable parts of the DATR language which do not contribute to its expressive capabilities and thus are dispensible. We will show that everything what can be expressed with the standard DATR language can also be achieved with a rather restricted subset of DATR which we will refer to as *local path DATR (LDATR)* throughout the present paper. LDATR has no crucial notion of *node* and works without global inheritance. Thus, LDATR is nothing more but the local path inheritance mechanism of standard DATR.

[1]See Cahill [1993], Gibbon [1992], Gazdar [1992], and Kilbury [1992] for recent DATR applications in these areas. An informal introduction to DATR is given in Gazdar [1990]. The standard syntax and semantics of DATR is defined in Evans & Gazdar [1989a, 1989b]. Implementation issues are discussed in Gibbon & Ahoua [1991], Jenkins [1990], Gibbon [1993], and Drexel [1994]. Moser [1992a, 1992b, 1992c, 1992d] provides interesting insights into the formal properties of DATR (see also the DATR representations of finite state automata, different kinds of logics, register operations etc. in Evans & Gazdar [1990]). Andry et al. [1993] describe how DATR can be used in speech-oriented applications.

Chapter 2

Encoding Nodes as Path Attributes

We will investigate a proper subset of DATR which we will refer to as *Local Path DATR (LDATR)* in the present paper. A LDATR theory consists of a constant node NODE and a set of clauses with a local path at the left hand side and an atom, a local path or a possibly empty sequence of atoms or local paths at the right hand side:

LDATR BNF syntax (standard DATR compatible version):
⟨theorem⟩ ::= NODE:< {⟨atom⟩} > = {⟨atom⟩} .
⟨theory⟩ ::= NODE:⟨clause⟩{⟨clause⟩} .
⟨clause⟩ ::= < {⟨atom⟩} > == ⟨rhs⟩
⟨rhs⟩ ::= (⟨apseq⟩) | ⟨apseq⟩ | < ⟨apseq⟩ >
⟨apseq⟩ ::= {⟨ap⟩}
⟨ap⟩ ::= ⟨atom⟩ |< ⟨apseq⟩ >

The procedural semantics of LDATR is identical to the first and fourth rule of inference as defined in Evans & Gazdar [1989a].

In this section we reconstruct a substantial subset of standard DATR in terms of LDATR. This subset includes all DATR theories which do not presuppose the sixth rule of inference. The reconstruction is hence restricted to DATR theories without quoted nodes. We define translation rules which, for an arbitrary standard DATR theory θ_1 without quoted nodes, yield a LDATR theory θ_2 which is weakly equivalent in the sense that each theorem of θ_1 has exactly one counterpart in θ_2. In particular we will show that a version of LDATR which includes DATR variables is moreover strongly equivalent to standard DATR without global nodes. By 'strongly equivalent' we mean:

- there is a one-to-one correspondence between the *theorems* of θ_1 and θ_2, respectively
- there is a one-to-one correspondence between the *clauses* of θ_1 and θ_2, respectively

- there is a one-to-one correspondence between the *steps of inference* which yield a theorem in θ_1 and the steps of inference which yield its counterpart in θ_2, respectively

2.1 Nodes and Attributes

DATR paths are (sometimes implicitly) associated with two different kinds of node:

- *local* nodes and
- *global* nodes

If a local path occurs without a node at the right hand side of a DATR clause it is implicitly associated with the left hand side node. Quoted paths are implicitly associated with the current global node. Nodes are not essentially different from attributes, both are atomic symbols. Thus, we can redefine local and global nodes in terms of path attributes[1]. We reserve the first and the second attribute of each path for the global and the local node, respectively. From a node-path pair N:$< a_1 \dots a_k >$ with a global node GN we thus can derive the path $< GN\ N\ a_1 \dots a_k >$. As long as there is no specific query the global environment is uninstantiated. In order to be prepared for queries to all nodes $N_1 \dots N_n$ of a given theory we have to map the left hand side N_i:$< a_1 \dots a_k >$ of each DATR sentence onto n paths which differ with respect to the first attribute, i.e. w.r.t. the global node. This mapping yields:

$<\mathrm{N}_1\ \mathrm{N}_i\ \mathrm{a}_1 \dots \mathrm{a}_k>$
$<\mathrm{N}_2\ \mathrm{N}_i\ \mathrm{a}_1 \dots \mathrm{a}_k>$
$\dots$
$<\mathrm{N}_n\ \mathrm{N}_i\ \mathrm{a}_1 \dots \mathrm{a}_k>$

A more readable notation can be achieved by using DATR *variables* (cf. Jenkins [1990]) instead of explicit attribute-encoded global nodes[2]. Provided there is a variable declaration

vars $node: $N_1\ N_2 \dots N_n$.

we can map a node-path pair N:$< a_1 \dots a_k >$ onto the single path

< $node N $\mathrm{a}_1 \dots \mathrm{a}_k$ >.

So far, we have given the translation rule for mapping left hand side node-path pairs onto local paths. For translating the right hand sides several rewrite rules for the different DATR objects which may occur at the *right* hand side are needed. For a given DATR clause with a left hand side path $< a_1 \dots a_n >$ at a node N1, the right

[1]The idea of encoding the local node as the first attribute of a path is due to Petra Naerger and Ingrid Renz. According to the usual DATR notation conventions nodes start with a capital letter, and atoms either start with minor letter or are enclosed in single quotes. In this paper we will omit these quotes in unambigous cases.

[2]Since variables are not part of the standard DATR language (though there are a couple of DATR implementations which support them) we regard them as *meta*-variables ranging over LDATR attributes (and *not* as a language primitive of LDATR) for the purpose of showing that LDATR is weakly equivalent to standard DATR. In section 4 we will discuss the expressive capabilities of a version of LDATR which includes variables as a genuine language primitive.

hand side objects are translated as follows:

DATR→LDATR translation rules (preliminary version)

1. **empty sequence:** $\longrightarrow \epsilon$

2. **atomic values:** atom $\longrightarrow$ atom

3. **node-path pairs:** $\mathrm{N2}{:}<b_1 \ldots b_m> \longrightarrow <\$\mathrm{node}\ \mathrm{N2}\ b_1 \ldots b_m>$

4. **node:** $\mathrm{N2} \longrightarrow <\$\mathrm{node}\ \mathrm{N2}\ a_1 \ldots a_n>$

5. **path:** $<b_1 \ldots b_m> \longrightarrow <\$\mathrm{node}\ \mathrm{N1}\ b_1 \ldots b_m>$

6. **quoted node-path pair:** $"\mathrm{N2}{:}<b_1 \ldots b_m>" \longrightarrow <\mathrm{N2}\ \mathrm{N2}\ b_1 \ldots b_m>$

7. **quoted path:** $"<b_1 \ldots b_m>" \longrightarrow <\$\mathrm{node}\ \$\mathrm{node}\ b_1 \ldots b_m>$

The following example illustrates the translation rules. Consider the standard DATR theory, below:

Standard DATR:

```
House:
   <> == Noun
   <root> == house.

Sheep:
   <> == Noun
   <root> == sheep
   <affix plur> == .

Foot:
   <> == Sheep
   <root> == foot
   <root plur> == feet.

Noun:
   <orth> == "<root>" "<affix>"
   <affix sing> ==
   <affix sing gen> == s
   <affix plur> == s.
```

The application of the translation rules, defined above, yields the following LDATR theory (the numbers of the translation rules applied, are given as a comment):

```
# vars $node: 'House' 'Sheep' 'Foot' 'Noun'.
```

```
NODE:
  <$node 'House'> == <$node 'Noun'>                                    % 4
  <$node 'House' root> == house                                        % 2
  <$node 'Sheep'> == <$node 'Noun'>                                    % 4
  <$node 'Sheep' root> == sheep                                        % 2
  <$node 'Sheep' affix plur> ==                                        % 1
  <$node 'Foot'> == <$node 'Sheep'>                                    % 4
  <$node 'Foot' root> == foot                                          % 2
  <$node 'Foot' root plur> == feet                                     % 2
  <$node 'Noun' orth> == <$node $node root> <$node $node affix> % 7
  <$node 'Noun' affix sing> ==                                         % 1
  <$node 'Noun' affix sing gen> == s                                   % 2
  <$node 'Noun' affix plur> == s.                                      % 2
```

The following traces illustrate that these two theories not only define equivalent theorem sets, but also equivalent *derivations* for the corresponding theorems:

Standard DATR sample trace:

```
Foot:<orth plur>
Sheep:<orth plur>
Noun:<orth plur>
Foot:<root plur>  Foot:<affix plur>
feet              Sheep:<affix plur>

THEOREM: Foot:<orth plur> = feet.
```

LDATR sample trace:

```
NODE:<'Foot' 'Foot' orth plur>
NODE:<'Foot' 'Sheep' orth plur>
NODE:<'Foot' 'Noun' orth plur>
NODE:<'Foot' 'Foot' root plur>   NODE:<'Foot' 'Foot' affix plur>
feet                             NODE:<'Foot' 'Sheep' affix plur>

THEOREM: NODE:<'Foot' 'Foot' orth plur> = feet.
```

Chapter 3

Global Paths

We have illustrated how both local and global nodes can be represented as path attributes, which is sufficient for developing LDATR representations for many, but not for all standard DATR theories, because we have ignored quoted nodes, so far, i.e. sentences such as e.g.:

```
N:<b> == "N1".
```

This kind of DATR statement means that the current global path is queried at the node N1 and that N1 is the global node for the further evaluation of this query. In order to achieve an LDATR representation of quoted nodes we have to solve the following problems:

- How can global paths be represented in terms of local paths alone?
- How can global paths be accessed?
- How can they be modified?

The problem of representing global paths in terms of local paths is essentially different from the problem of representing local and global nodes as path attributes, since nodes can be referred to by their position, while a global path has an arbitrary length and thus cannot be referred to by a fixed position.

We will use particular techniques for representing, accessing, and manipulating arbitrary sub-paths which we borrow from Gibbon (1989) and Moser (1992a). For the purpose of defining global paths in terms of local paths we first define some basic functions in LDATR which are useful tools for the definition of more complex path processing procedures. We indicate LDATR attributes which represent function names by asterisks; the function name is the first attribute of a path and the other attributes (including the extensional suffix, provided there is any) represent the arguments. If the function is complex (i.e. if it calls other functions) the right hand side can be regarded as the body of the function, otherwise the right hand side is the value. As their names mnemonically indicate, we presuppose that the variables we use below are defined for particular domains: the variable $node ranges over those attributes of a LDATR theory which correspond to the nodes of the original standard DATR theory, and $attribute ranges over all attributes of the

LDATR theory, except "!".

copy is a function that maps a path onto a list:

```
<*copy*> == ()
<*copy* $attribute> == ($attribute <*copy*>)
```

copy_until is a partial function that evaluates to the list of attributes which are the prefix of the extensional suffix up to the first occurence of a reserved attribute (e.g. "!"):

```
<*copy_until* !> == ()
<*copy_until* $attribute> == ($attribute <*copy_until*>)
```

copy_from uses the special attribute '!' and returns the suffix of the argument path after the first occurrence of "!":

```
<*copy_from* !> == <*copy*>
<*copy_from* $attribute> == <*copy_from*>
```

The function *cut_out* selects the first occurrence of a substring of a path starting and ending with the delimiter "!":

```
<*cut_out*> == <*copy_until* <*copy_from*>>
```

The sentence

```
<$node> == <$node $node <*copy*> ! <*copy*> !>
```

allows for convenient queries. It maps an initial query path of the form

$<\text{NODE } a_1 \ldots a_n>$

onto

$<\text{NODE NODE } <\text{*copy* } a_1 \ldots a_n> \; ! \; <\text{*copy* } a_1 \ldots a_n> \; ! \; a_1 \ldots a_n>$

The evaluation of the embedded paths yields finally the internal representation of the local and global query evaluation environments:

$< \text{NODE NODE } a_1 \ldots a_n \; ! \; a_1 \ldots a_n \; !a_1 \ldots a_n >$

This internal representation differs from the original query in that the node and the path are multiply represented, respectively. The first occurence of NODE represents the global node and the second the local node; the first occurrence of $a_1 \ldots a_n$ represents the local path and the second the global path. The first occurrence of "!" indicates the boundary between the local and the global path and the second occurrence indicates the end of the global path, which has to be separated from the following third occurrence of the extensional suffix. This third occurence of the path is of no significance, but it is impossible to avoid it.

Using the basic functions, defined above, it is trivial to construct functions for accessing the global path and the extensional suffix, respectively:

```
<*gp*> == <*cut_out*>

<*es*> == <*copy_until*>
```

Notice that definitional sentences in LDATR always start with *two* nodes. Thus, a query can easily by recognised (and distinguished from definitional sentences).

Chapter 4

Mapping Standard DATR onto LDATR

In this section we revise the set of translation rules defined in section 2. The revision yields a set of rules which, if applied to an arbitrary standard DATR theory θ_1, yield a LDATR theory θ_2 such that for each theorem of θ_1 of the form $N :< a_1 \ldots a_n >= V$ there is a theorem of θ_2 of the form NODE:$<' N' a_1 \ldots a_n >= V$. The theory θ_2 defines of course additional theorems which have no counterpart in θ_1: these are those theorems which derive from the 'built-in functions', as e.g. *copy*, *gp*, etc.

Basically, a LDATR representation derived from a standard DATR theory has three parts:

1. declarations for the variables $node and $attribute: Let be $\mathbf{N} = \{N_1, \ldots, N_k\}$ the set of nodes of θ_1 and $\mathbf{A} = \{a_1, \ldots, a_n\}$ the set of path attributes that occur in θ_1. Then θ_2 contains the variable declarations, below:

   # vars $node: '$N_1$' '$N_2$' ...'$N_k$'.
   # vars $attribute: $a_1 \ldots a_n$ 'N_1' 'N_2' ...'N_k'.

2. the fixed set of path manipulating functions defined in LDATR (*gp*, *copy* etc.)

3. the LDATR representations of the original clauses

Each left hand side N1:<$a_1 \ldots a_n$> in θ_1 corresponds to the path <$node 'N1' $a_1 \ldots a_n$> in θ_2.

There are eight different kinds of objects which may occur as at the right hand side of a standard DATR clause. The translation rules, given below, map atoms onto themselves and all other objects onto local paths:

DATR⟶LDATR translation rules (final version)

1. **empty sequence:** $\longrightarrow \epsilon$

2. **atomic value:** atom $\longrightarrow$ atom.

3. **node-path pair:** N2 :$< b_1 \ldots b_m > \longrightarrow <$\$node 'N2' $b_1 \ldots b_m >$.

4. **node:** N2 $\longrightarrow <$ \$node 'N2' $a_1 \ldots a_n >$.

5. **path:** $< b_1 \ldots b_m > \longrightarrow <$ \$node 'N1' $b_1 \ldots b_m >$.

6. **quoted node-path pair:** "N2 :$< b_1 \ldots b_m >$ " $\longrightarrow$
   $<$ 'N2' 'N2' $b_1 \ldots b_m <*es*>$! $b_1 \ldots b_m <*es*>$! $>$.

7. **quoted node:** "N2" $\longrightarrow <$ 'N2' 'N2' $<*gp*>$ $<*es*>$! $<*gp*>$! $>$.

8. **quoted path:** "$<b_1 \ldots b_m>$" $\longrightarrow$
   $<$\$node \$node $b_1 \ldots b_m <*es*>$! $b_1 \ldots b_m <*es*>$!$>$.

We illustrate the mapping from standard DATR onto LDATR by an example theory which includes all standard DATR rules of inference, sequences, defaults, overriding, and multiple embedding.

Standard DATR:

```
N1: <a> == N2:<b> <b>    % II, IV; sequence
    <b> == 1             % atomic
    <>  == N2.           % III; default
N2: <b> == n2b
    <c> == "<b>" "N3:<1>" <<"<b>">> "N3"
 % V, VI, VII; multiple embedding
    <1> == 2
    <2> == 3.
N3: <1> == "<c>"
    <c> == 4.
```

Sample Theorems:

```
  N1:<a> = n2b 1.
  N1:<b> = 1.
  N1:<c> = 1 4 3 4.
```

LDATR:

```
# vars $node: 'N1' 'N2' 'N3' .
# vars $attribute: a b c 1 2 'N1' 'N2' 'N3'.

NODE:
   <$node 'N1' a> == <$node 'N2' b> <$node 'N1' b>
   <$node 'N1' b> == 1
   <$node 'N1'> == <$node 'N2'>
   <$node 'N2' b> == n2b
```

```
   <$node 'N2' c> ==
<$node $node  b <*es*>  ! b <*es*> !>
<'N3' 'N3' 1 <*es*>  ! 1 <*es*> !>
<$node 'N2' <$node 'N2'
   <$node $node b <*es*> ! b <*es*> !>>>
<'N3' 'N3' <*gp*> <*es*> ! <*gp*> !>
   <$node 'N2' 1> == 2
   <$node 'N2' 2> == 3
   <$node 'N3' 1> == <$node $node c <*es*> ! c <*es*> !>
   <$node 'N3' c> == 4

   <*copy*> ==
   <*copy* $attribute> == $attribute <*copy*>
   <*copy_until* !> ==
   <*copy_until*  $attribute> == $attribute <*copy_until*>
   <*copy_from* !> == <*copy*>
   <*copy_from* $attribute> == <*copy_from*>
   <*cut_out*> == <*copy_until* <*copy_from*>>
   <$node> == <$node $node <*copy*> ! <*copy*> ! >
   <*gp*> == <*cut_out*>
   <*es*> == <*copy_until*> .
```

Sample Theorems:

```
NODE:<'N1' a> = n2b 1.
NODE:<'N1' b> = 1.
NODE:<'N1' c> = 1 4 3 4.
```

We now summarize the results:

1. LDATR (with or without variables) is weakly equivalent to standard DATR in the sense that for an arbitrary standard DATR theory θ_1 a LDATR theory θ_2 can be constructed such that θ_2 defines an equivalent theorem for each theorem of θ_1.

2. The LDATR translation of a standard DATR theory has in the worst case the size of $n^2 + 10$ clauses (where n is the number of clauses of the original standard DATR theory and 10 is the number of the clauses for the definition of the 'built-in functions'). This means, of course, that LDATR does not capture all generalisations of the underlying standard DATR theory explicitly. Thus LDATR is not strongly equivalent to standard DATR.

3. LDATR *with* variables is strongly equivalent to the set of DATR theories without quoted nodes and captures all generalisations of the underlying standard DATR theory with the same degree of explicitness.

Chapter 5

LDATR as a Modelling Convention for Standard DATR

In the previous sections of the present paper we have shown that standard DATR is weakly equivalent to LDATR, and that a version of LDATR with variables is moreover strongly equivalent to a substantial subset of standard DATR. These results might be of interest for the theoretical understanding of DATR, but do not necessarily contribute directly to the primary aim of (better and easier) lexicon development. In this section we sketch some modelling strategies which illustrate that the reconstruction of global inheritance and nodes in terms of local paths allows for flexible modifications of the basic concept of global inheritance which might turn out to be useful for the task of practical lexicon development.

5.1 Constraints on Global Accessibility

As a welcome side effect of the attribute encoding of global nodes we now can easily differentiate between nodes which are intended to be globally accessible and those nodes which are not ('abstract' or 'hidden' nodes) by including only those nodes in first position which we intend to be accessible by a query or by global inheritance[1].

Constraints on global accessibility are not only useful for explicitly representing linguistically motivated distinctions between nodes which represent classes and nodes which represent instances. They may be also useful in order to avoid non-terminating queries. Consider the following example theory:

```
NOUN:<orth> == ("<orth stem>" "<orth ending>").
```

[1] This option allows for expressing explicitly what is implicitly presupposed by many DATR representations which have these two partitions of nodes, e.g. ILEX style representations (cf. Gibbon 1992). Furthermore, the search space for computing reverse queries (cf. Langer 1994) with derivations which involve quoted paths is reduced from *all* nodes of a given theory to those nodes which would be suitable query nodes. Since backwards resolution of quoted paths is one of the most expensive operations in reverse query evaluation this search space reduction is likely to decrease the complexity of reverse queries to a considerable degree.

```
HOUSE:
     <> == NOUN
     <orth stem> == house
     <orth ending> ==
     <orth ending plur> == s.
```

Queries like `NOUN:<orth>` do not terminate. In the above DATR theory (and in many other theories), references to the global node via quoted paths are appropriate only if the current global node is different from the local node. Since this constraint on global accessibility is not explicit in the given theory, it is left to the user (or to the lexical access procedure of an NLP system) to avoid 'silly queries'. A LDATR version (with variables) of a standard DATR theory defines the domain of accessible global nodes explicitly as a variable declaration. For the DATR theory, above, the variable declaration of the corresponding LDATR translation is

```
# vars $node: 'NOUN' 'HOUSE'.
```

If the former node `'NOUN'` is removed from the list of permitted instances of the variable `$node` the distinction between lexeme-denoting nodes and generalisation-denoting nodes is represented explicitly. The revised LDATR version still defines the same set of theorems. But the query `NOUN:<orth>` yields immediately a failure without getting lost in a cycle.

5.2 Multiple Global Environments

In some situations it is useful to have the option to access the very initial global environment of a query at a step of inference where the global environment has already been modified. Attribute encoding of global nodes allows for this option. The basic idea is simply to store the initial node twice as a path attribute:

```
<'GN' 'GN' 'LN' a1 ... an>
```

The first global node may be modified in the usual way, while the second global node serves as a read-only storage of the very initial global node.

Chapter 6

Conclusions

We have shown that standard DATR has some redundancies, i.e. there are particular data types and operations which turn out to be "syntactic sugar". We do not doubt that this syntactic sugar (i.e. nodes and global inheritance) may have an important heuristic value. Nevertheless, these elements of the language are dispensable from a formal point of view, because they do not affect its expressive capabilities. The result of the "no-sugar diet" is a slim subset of DATR called *LDATR*. Local Path DATR has been shown to be

- DATR-equivalent and hence Turing-equivalent
- less redundant and simpler w.r.t. syntax and semantics
- more flexible (LDATR encoding of nodes as path attributes allows for defining constraints on global accessibility and multiple global environments)

Chapter 7

References

Andry, François, Norman M. Fraser, Scott McGlashan, Simon Thornton & Nick J. Youd [1992]: Making DATR Work for Speech: Lexicon Compilation in SUNDIAL. In: *Computational Linguistics* Vol. 18, No. 3, pages 245-267.

Cahill, Lynne J. [1993]: Morphonology in the Lexicon. In *Sixth Conference of the European Chapter of the Association for Computational Linguistics*, pages 87-96, 1993.

Drexel, Guido [1994]: Repräsentation hierarchischer Lexika: DATR in einer objektorientierten Umgebung. Ms. University of Bielefeld.

Evans & Gazdar [1989a]: Roger Evans & Gerald Gazdar: Inference in DATR. In *Fourth Conference of the European Chapter of the Association for Computational Linguistics*, pages 66-71, 1989. Also in: Evans & Gazdar [eds.] [1990].

Evans & Gazdar [1989b]: The Semantics of DATR. In Anthony G. Cohn, editor, *Proceedings of the Seventh Conference of the Society for the Study of Artificial Intelligence and Simulation of Behaviour*, pages 79-87,London 1989, Pitman/Morgan Kaufmann. Also in: Evans & Gazdar [eds.] [1990].

Evans & Gazdar [eds.] [1990]: The DATR Papers. Brighton: University of Sussex Cognitive Science Research Paper CSRP 139, 1990.

Gazdar, Gerald [1992]: Paradigm Function Morphology in DATR. In: L. J. Cahill & Richard Coates [eds.]: Sussex Papers in General and Computational Linguistics: Presented to the Linguistic Association of Great Britain Conference at Brighton Polytechnic, 6th-8th April 1992. Cognitive Science Research Paper (CSRP) No. 239. University of Sussex, 1992, pages 43-54.

Gibbon, Dafydd [1989]: register.dtr. In: Evans & Gazdar [eds.] [1990], pages 133-134.

Gibbon, Dafydd [1992]: ILEX: A linguistic approach to computational lexica. In:

Ursula Klenk [ed.]: Computatio Linguae. Aufsätze zur algorithmischen und quantitativen Analyse der Sprache, pages 32-53.

Gibbon, Dafydd [1993]: Generalised DATR for flexible access: Prolog specification. English/Linguistics Occasional Papers 8. University of Bielefeld.

Gibbon, Dafydd & Firmin Ahoua [1991]: DDATR: un logiciel de traitement d'héritage par défaut pour la modélisation lexical. Chiers Ivoriens de Recherche Linguistique (civl) 27. Université Nationale de Côte d'Ivoire. Abidjan, 1991, pages 5-59. Also published as: English/Linguistics Interim Report No. 4. University of Bielefeld.

Jenkins, Elizabeth A. [1993]: Enhancements to the Sussex Prolog DATR Implementation. In: Evans & Gazdar [eds.] [1990], pp. 41-61.

Kilbury, James [1992]: Paradigm-Based Derivational Morphology. In: Günther Görz [ed.]: KONVENS 92. Springer, Berlin etc. 1992, pages 159-168.

Langer, Hagen [1994]: Reverse Queries in DATR. [To appear in: Proceedings of COLING-94].

Moser, Lionel [1992a]: DATR Paths as Arguments. Cognitive Science Research Paper CSRP 216, University of Sussex, Brighton.

Moser, Lionel [1992b]: Lexical Constraints in DATR. Cognitive Science Research Paper CSRP 215, University of Sussex, Brighton.

Moser, Lionel [1992c]: Evaluation in DATR is co-NP-Hard. Cognitive Science Research Paper CSRP 240, University of Sussex, Brighton.

Moser, Lionel [1992d]: Simulating Turing Machines in DATR. Cognitive Science Research Paper CSRP 241, University of Sussex, Brighton.