ESF Eurotype

Central Coordination
Dept. of English
Free University of Berlin
Gosslerstraße 2-4
D-1000 Berlin
Germany

ESF Office
1, quai Lezay Marnésia
F-67000 Strasbourg
France

Working Paper 3

A Database System for Language Typology

Dik Bakker
Anna Siewierska

Theme Group 2: Constituent Order
Coordination: Anna Siewierska

Anna Siewierska
Department for General Linguistics
University of Amsterdam
Spuistraat 210
1012 VT Amsterdam
The Netherlands

phone: +31 20 5253860
fax: +31 20 5253052
e-mail: annas@alf.let.uva.nl

Dik Bakker
Department of Computational Linguistics
University of Amsterdam
Spuistraat 134
1012 VB Amsterdam
The Netherlands

+31 20 5252070

dik@alf.let.uva.nl

1. Introduction

One of the means of collecting cross-linguistic data, an activity which by virtue of the nature of the EUROTYP project we are all currently involved in, is via a questionnaire. Most linguistic questionnaires are aimed at eliciting both primary data, i.e. actual language utterances, and analytical data, i.e. statements about the grammatical characteristics of languages. In the course of the last year we have developed a scheme for processing, storing and analyzing analytical data. The scheme has been elaborated on the basis of a questionnaire on word order compiled by the constituent order group. However, significantly, it is not dependent on the word order or any other linguistic domain.

The proposed scheme is intended to facilitate computer data entry and the analysis of linguistic data by means of generally available and specially developed programs. It constitutes a powerful analytical tool which can be easily applied to any linguistic domain. In order to use the system all that is required is that the linguistic domain under investigation be structured in a particular variable - value format and transformed into a normal computer readable textfile. Once this is accomplished, all the analytical procedures can be applied automatically without further technical demands on the linguist. The linguist "only" needs to apply her/his linguistic knowledge and expertise in reducing the number of possibilities open to the computer programs or alternatively in selecting from a body of potentially relevant linguistic facts those which are of actual interest. A global outline of the - fully interactive - computer system is depicted in figure 1.

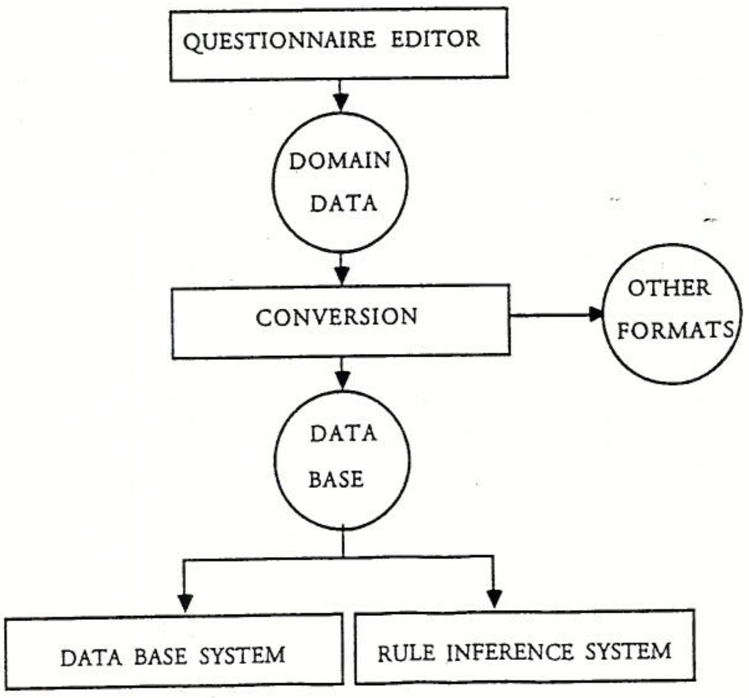


figure 1

Reading this scheme from top to bottom, there is first a module for entering the questionnaire data into the computer according to a special format to be discussed below. A second module integrates these data into a centralized data base, that may contain data on other linguistic domains than the one of the user. With that same module one may transform the format of the data to that of other, often used computer programs, e.g. for statistical analysis. With a third program, the centralized data base, containing the user's data, may be searched. With the bottom module, the data may be analyzed according to non-statistical methods, such as the determination of implications, and other strategies of linguistic data analysis.

Since the structure of the linguistic domain in terms of variables and values constitutes the input to the analytical procedures, before presenting the various modules of which the system is comprised, we will first describe the required format. After introducing, in section 2, some of our innovations to the standard variable - value design, in section 3 we will describe the formal conventions used in the textfile representation of this design. Section 4 will constitute the body of the paper. It will begin with a brief presentation of the overall organization of the computerized data base and the set of computer programs supporting it. The following subsections will describe in more detail the function of each of the modules of the overall system. Special attention will be devoted to the analytical possibilities incorporated into the inferencing program which has been developed on the basis of the cumulative achievements of typological research over the last decades.

2. The structuring of variables and values

Any linguistic dimension may constitute a **variable**. For instance when dealing with word order one is likely to set up variables such as: the order of the subject, object and verb in declarative clauses; the position of the question word in wh-questions; the location of the negative marker; the order of sequences of nominal modifiers etc. **Values** are fixed points on the linguistic dimension represented by the variable. Typically each variable has a finite set of values. Thus in the case of the variable *order of the subject, object and verb in declarative clauses* the values could be any of the items in (a) through (f) below.

(1) V1 Declarative order of S, V & O
- a. SVO
- b. SOV
- c. VSO
- d. VOS
- e. OSV
- f. OVS

Of the four types of variables that are usually distinguished in data analysis - nominal, ordinal, interval and ratio - the lowest on the scale, nominal variables, are by far the most common in linguistic research. (1) is a good example of a nominal variable [see note 1].

The structuring of a linguistic domain in terms of variables and values can be achieved in numerous ways. What constitutes a variable and what a value is purely up to the linguist, and the theoretical framework that has been adopted. For instance, instead of the variable *order of the subject, object and verb in declarative clauses*, one could have a series of variables stipulating the possibility of occurrence of each of the six transitive orders, with *yes/no* as values. Or instead of terms such as subject and object, one could use semantic roles, case marking, constituent structure categories etc.

When working with a questionnaire part of the structuring of a linguistic domain with respect to variables and values is already incorporated into the questionnaire. The questions in the questionnaire correspond to the variables, in some way or another, and the answers to the questions reflect the possible values. However, given the impossibility of foreseeing all the linguistic dimensions that are relevant for a particular domain in compiling a questionnaire and the necessity of taking into consideration the psychological impositions on the informants answering it, there is not always a one to one relationship between the questions in a questionnaire and the variables that one finally sets up to store and process the results. Therefore a certain amount of restructuring is typically required in the establishing of a variable - value format appropriate for data storage.

In designing a scheme for the capturing of linguistic data we have tried to cater for the specificity of language data and the analytical methods used by linguists. We have thus enriched the simple variable - value format by several conventions which together enable the representation of language facts in a way closely corresponding to the actual practice of linguists. Each of these conventions will be discussed in turn below [2].

2.1 Multiple values

In a usual variable - value scheme consisting of a variable with a set of values, only one of the values may be chosen for each language. This is, obviously, much too restrictive for the representation of linguistic data. Adherence to such a system would not only lead to a proliferation of variables, but more importantly to the loss of significant generalizations. In the system that we have developed any language can have multiple values for any variable. Thus, for example, for the variable *form of impersonalizing strategies* most languages select at least three of the values in (2).

(2) V32.1 impersonalizing strategies
- a. ImpOne
- b. ImpMan
- c. ImpPeople
- d. ImpSomeone
- e. Imp2ndsg
- f. Imp3rdsg
- g. Imp3rdpl
- h. Imp1stpl
- i. ImpRefl
- j. ImpSpecial
- k. ArgOmission
- l. ImpPass

There is no principled upper bound on the number of values that a language may take for a given variable. So far the maximum number of values assigned to one language for a variable in the word order data base is eleven. The language which required so many values was Lezgi and the variable in question was *adverbial cases distinguished*. We have found that most of the variables in the word order data base require multiple values for some language or another.

2.2 Compound values

It is sometimes the case that two values of which one is relevant for language X and the other for language Y must co-occur in a third language Z. Such a situation can be handled by means of syntagmatic conditions (see section 2.5). Alternatively, it can be dealt with by compounding the two values in question into a new complex value. When more than two values are involved and they can apply in several different combinations, this second alternative may be the more attractive. Consider, for instance, the case of the pragmatic restrictions underlying OSV order. On the basis of the responses to the word order questionnaire the major pragmatic conditions underlying the OSV pattern in the languages considered so far are as shown in (3).

(3)
a. Otopic Sfocus
b. Otopic Vfocus
c. Otopic Xfocus
d. Ofocus
e. Sfocus
f. Ocontrastive
g. Scontrastive
h. Ocontrastive Sfocus
i. Ocontrastive Xfocus

The above pragmatic conditions can be captured directly by means of a set of values corresponding to the list in (3). On the other hand, one could also have a reduced set of values, as in (4), and whenever necessary apply the option of compounding the simple values into complex ones.

(4)
a. Otopic
b. Sfocus
c. Vfocus
d. Xfocus
e. Ocontrastive
f. Scontrastive

The compounding option is particularly handy in the early stages of processing questionnaires. It is not always possible, or perhaps even not desirable, to postpone the task of restructuring a linguistic domain in terms of a variable - value format until all the completed questionnaires come in. Therefore often one does not have a comprehensive list of values for each variable to start off with. The possibility of compounding simple values into complex ones, is one of the "painless" means of expanding the value list.

2.3 Complex values

We have just seen that values may be combined. There may also be "dissected" so to speak. Many of the values used in the word order data base are complex, i.e. they consist of two or more linguistically identifiable segments. Thus the value SVO can be analyzed into S, V and O, the value singular-dual-plural can be split up into singular, dual and plural etc. A formal method has been developed which internally analyzes such complex values, provided that their sub-parts are predefined, and that a set of rules is available for their structure. This allows one to generalize over values and regroup them onto sub-sets such as VO/OV, SV/VS, Vinitial, singular/non-singular etc. See section 4.4.8.1 for more on this.

2.4 Scales for values

When a given language selects more than one-value for a particular variable, it may be the case that the values in question can be assigned a preferential, dialectal, stylistic or other ranking. For example, many languages display several possibilities as far as the placement of adverbials of setting is concerned. Yet in most the initial or final position in the utterance is clearly favoured above clause internal placement. To give another example, in standard Finnish it is not possible to question the subject of a subordinate clause. In colloquial Finnish, on the other hand, such questions are grammatical. Yet another case in point, in English *one* may be used as an indefinite pronoun. This usage is, however, associated with the formal language.

Scalar values reflecting preferences or dispreferences such as the above may be directly appended to values proper, leading in fact to complex values. In Matthew Dryer's data base, for example, a preferential order is denoted by values in upper case and a dispreferred one by lower case (see Dryer 1987). One can imagine various other systems. However, since more often than not many combinations are possible between variable values and scale values both theoretically and practically, directly combining the two would lead to obscuring the values proper. This in turn would complicate any analysis of the data by rendering necessary a large number of reinterpretation and recoding operations. The alternative that we have adopted is that of separate scale values which may be treated independently of the variable values themselves. The advantage of this is that the scale values can be taken into consideration only when they are considered to be relevant, for example, when determining details of combinatorial possibilities and not the very existence of those possibilities in a language.

In the word order data base, so far the following scales have been used:

```
(5)     Preference=[oblig, pref, non_pref, rare]
        Restrictive=[fav, restr]
        Style=[colloqial, literary, dialectic, formal, spoken, written]
        Obligatory=[obl, opt]
```

The scales can be applied in combination. Thus, for instance, phrasal discontinuity in Russian has a value on the stylistic scale as colloquial, and on the preferential scale as pref(erred).

2.5 Conditions on variables and values

Perhaps the most far reaching innovation incorporated into the described variable - value scheme from the point of view of data storage is the possibility of stating conditions on variables and values. Two types of conditions have been catered for, paradigmatic conditions on variables, and syntagmatic conditions on values. The paradigmatic conditions capture the relevance of particular variables for a given language. In doing so they reflect the potential dependency relations obtaining between variables and thus help to determine what may and what may not be in a grammar of a language. To give a simple example, in the word order data base, if a language has no articles, then the next potentially relevant variable is *the form of the demonstrative*. This is a reflection of the fact that in many languages articles have evolved out of demonstratives, and that often demonstratives fulfill some of the functions of articles.
Paradigmatic conditions may also be used to capture a relationship between a

particular variable and the set of values in another variable, restraining it to a subset, in case of some specific score on the higher variable.

Syntagmatic conditions describe the circumstances under which the phenomenon coded by the variable concerned occurs in a language. The need for expressing such conditions is obvious: no matter how finely we structure our linguistic domain by means of variables and values, we cannot hope to capture in just this manner all the existing distinctions found in languages.

Syntagmatic conditions can refer to any type of linguistic facts: lexical, semantic, categorial, morphological, structural, pragmatic, textual etc. The only restriction that has been imposed on the syntagmatic conditions is that they must be stated in terms of boolean expressions over variable-value pairs [3]. In most cases this can be done quite easily. By way of illustration, let us go through a couple of actual examples.

In Upper Sorbian yes/no questions may optionally contain either an unbound question particle, namely *hac* or the clitic *li*. Given the variable and values specified in (6), the nature of the relevant particles can be stated as a syntagmatic condition on each of the two values as shown in (6).

```
(6)   V14.2 boundness of Q-particle in yes/no questions
      a.     NB     [opt,(Qprt=hac)]
      b.     Cl     [opt,(Qprt=li)]
```

The factors underlying the use of the so called Saxon genitive which preferentially occurs only with short animates may serve as an example of a potential semantic condition. E.g.

```
(7) V21.1 structure genitive
      a.     Adp
      b.     GenSuff      [pref, (Semantic_Category_Gen=animate AND
                                   Weight_Gen=short)]
```

In (8) there is a complex syntagmatic condition specifying that the postposition of an adjective in English is obligatory when the adjective takes its own complement (e.g. *man proud of his son)*, but optional when such an adjective modifies an indefinite noun and is itself modified by an intensifer (e.g. *a very easy name to remember* vs *a name very easy to remember*).

```
(8) V20.5 order adjective N
      a.     AdjN
      b.     NAdj [[obl, (Structure_Adj=Adjcomp)],
                   [opt, (Structure_Adj=IntAdjcomp AND
                          Semantic_Category_NP=indef)]]
```

And finally in (9) we have two structural syntagmatic conditions accompanying the distinct pragmatic requirements underlying the occurrence of OSV order in Finnish.

```
(9)   V4.6p pragmatic restrictions OSV
      a.     OtopicSfocus (Pattern=XOSV)
      b.     OctrVfocus (Pattern=OSVX)
```

Just as scale values, syntagmatic conditions could be built into the value proper. However, given their often rather idiosyncratic nature, proliferation of variable values would be even greater than in the case of scales. When the syntagmatic conditions get too complex, it may be desirable to posit an additional variable or variables catering for the factors that reoccur in the syntagmatic conditions on some value in several of the languages.

For any variable, scales and syntagmatic conditions may be combined, as in the example (8) above. This provides a three-dimensional scheme for the representation of linguistic information, where the values on these dimensions - variables; scales; conditions - in themselves may be compound and complex.

Having outlined the possibilities that are available for the structuring of variables and values in our scheme, we now turn to the description of the formal representation of the format that has been adopted.

3. The formal structure of the DSL file

The variable - value format, which from now on will be referred to as the Domain Structuring Language (DSL), is intended to structure the data from a questionnaire, to drive computer programs for entering the data, and finally to serve as the input for analytical computer programs. In order to fulfill these functions, the data must be structured in a formal way. Therefore the following discussion will be somewhat technical.

A DSL file defining the characteristics of a linguistic domain begins with a specification of the domain under analysis and an enumeration of the scales and the type of missing values that will be used, as in (10) [4].

```
(10)  a. Word Order Data, Version 1.0  Nov91
      b. Scales:  Preference=(obligatory, preferred,
                              non_preferred, rare)
                  Restrictive=(fav, restr)
      c. Missing values:      (not_known, not_clear, not_present)
```

The body of a DSL file consists of a set of domain variable descriptors, where each descriptor denotes a separate variable. The structure of a specific domain variable descriptor is shown in (11).

(11) a. Variable name
     b. Reference
     c. Variable label
     d. Parameters
     e. Value set

The variable name is a short, unique indication for the variable for quick reference, not necessarily mnemonic. In the DSL file of the word order data base the variable names consist of the letter *V* (for variable) followed by a real number, e.g. V2.1, V14.2, V21.1 etc. The first digit of the number identifies the sub-domain to which the variable refers. For example, all variables beginning with V2. concern intransitive clauses, those beginning with V14. refer to yes/no questions, and those beginning with V21. involve possessive constructions. The second number is sequential. Needless to say, one can adopt any other means of naming variables.

The reference is an (optional) indication to which questions in the questionnaire the variable relates. The value of a variable may be derived from - the interpretation of the answers to - one or more questions or other information sources. Thus the reference is basically a convenience for checking the original source of the data in the questionnaire.

The variable label is a specification of the meaning of the variable, for instance, order of the possessor and possessed, inflectional cases on pronouns, existence of impersonal passive etc.

The parameters specify whether the variable in question will be characterized by means of scales, compound values, paradigmatic and/or syntagmatic conditions, and whether it is to be considered as multi-valued and/or open. The parameter open means that values not included in the value set (see below) can be added. It is important to note that whenever any of these options are to be activated for a variable, this must be made explicit independently for each variable.

The value set is the list of values for the variable. There are no a priori restrictions on the number of values for a given variable. Nonetheless if there should end up being as many values as there are languages one would be well advised to reconsider the viability of the variable in question.
Some of the above features of the variable descriptor are illustrated in (12).

```
(12)  V2.1
      (Q2.1 - Q2.4)
      Main bare intransitive orders
      parameters=(       multi=2, scale=Preference,
                         syntagmatic_condition),

      a. SiV
      b. VSi
```

The parameters given here should be interpreted as follows. 'multi=2' means that to a maximum of two values (in this case in fact: all) may be chosen from the list of predefined values. To any one of them a value from the scale 'preference' may be associated. The presence of the 'syntagmatic_condition' parameter allows a syntagmatic condition to be associated to any value chosen for the variable for some language. Recall that syntagmatic conditions are framed in terms of boolean expressions over variables and values as established in section 2.4. For variable V2.1 we may end up having the following complete value description for some specific language L:

```
(13)  V2.1= VSi - [(Preference=obligatory) ,
                  ((Pos1=expletive) OR (Pos1=adverb) OR
                  (Semantic_Category_S=indef))] / SiV
```

This should be interpreted as: in intransitives of L the subject follows the main verb as the obligatory order in case there is an expletive or adverb in the first position and also in case the subject is indefinite. In all other cases the order is subject-verb.

There is no paradigmatic condition or compound parameter in (13). The way paradigmatic conditions are represented is shown in (14), where the condition states that if the value *none* is chosen, i.e. if a language has no case marking on nouns, all the following variables concerning nominal case marking should be considered as not relevant, and the next variable to be regarded is V29.9, the variable pertaining to case marking of pronouns.

```
(14)  V29.8a
      (Q29.3, Q29.4)
      nominal cases
      parameters= (multi=10, syntagmatic_condition,
                        paradigmatic_condition [V29.8a=none > V29.9])
```

(15) illustrates the compound parameter which here specifies the possibilit though not the necessity, of combining any three values into a compound valu

```
(15) V28.1
      (Q28.1)
      agreement categories of the adjective
      parameters=(compound=4, multi=4, syntagmatic_condition)
      a.     no Adjective agreement
      b.     Number
      c.     Gender
      d.     Case
      e.     Definiteness
      f.     Class
```

Note that if for a given language there are syntagmatic conditions on any of the values, it may be preferable not to combine the particular values, unless the same syntagmatic conditions hold in each case.

When all the decided upon variables are given their appropriate variable descriptor format, the DSL file is complete. What now remains to be described is how such a DSL file can be used.

4. Computer programs for typological research

The DSL file is the backbone of a set of computer programs developed for entering, storing and retrieving analytical linguistic data, and performing analyses on them. The whole scheme takes the form given in figure 2. The scheme is interactive: all programs are run from the screen, in a dialogue with the user.

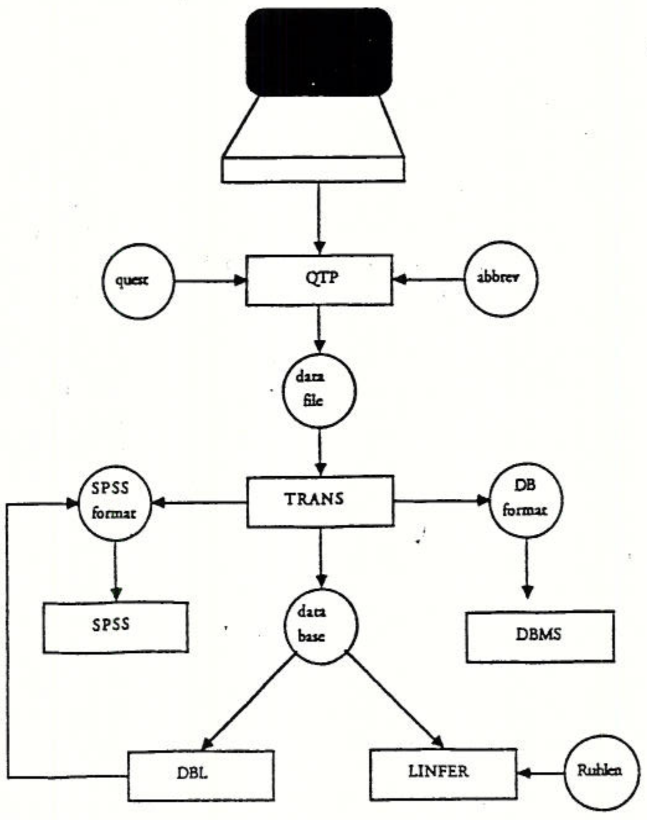


figure 2

QTP is a data entry program, to be used for entering the data coming in from questionnaires.
TRANS may be used to integrate the data stemming from a questionnaire on a certain subdomain into a central data base. It may also be used to transform these data such that they can be handled by some widely available applications.
DBL is the program that explores the central data base.
LINFER contains several methods of data representation and analysis often used in the field of language typology.
In the following sections, these programs will be discussed in more detail. This will be done from the perspective of their use in typological research. Technical details on them are to be found in appendix E.

4.1 The QTP data entry program

The QuesTionnaire Processor program (QTP) is meant as an instrument to enter questionnaire data into the computer in a systematic and controlled way, and store them in a domain dependent data base according to a unified format. The program itself is fully domain independent. It is applied to the previously constructed domain specific DSL file. The QTP program loads the DSL file, and checks its consistency [5]. Then questionnaire results for a user determined set of languages may be entered via keyboard interaction. When one starts the QTP program, after choosing the domain and a specific language for which there are, or should be, data in that domain, say Abxaz, the following menu is presented on the screen.

```
(16)        0. quit Abxaz
            1. options
            2. save changes to data base
            3. add new values
            4. correct existing values
            5. inspect values
```

One now may proceed to enter data via option 3. The program will prompt the user for values for domain variables in the hierarchical order in which they are defined in the DSL file. All parameters for a variable will be taken into consideration, as well as the values for any higher variables, that have already been specified for that language before, in the same session, or during a previous entry session, since they may play a role in paradigmatic conditions. Variable after variable will be presented in the following way:

```
(17) Language:    Abxaz
     Variable:    V18.2
     Reference:   (= Q18.2)
     Lable:       location of negative
     Synt Cond:   yes
     Maximum:     4 values
     a. initial
     b. final
     c. SVOneg
     d. postAux
     e. fusedwithAux
     f. preV
     g. postV
     z. other
```

The user may choose one value for the variable (in the case of a multi-valued variable: a number of different values to the maximum of the multi parameter) by typing the corresponding letter. In case of a scale being active, the user is prompted for an (optional) scale value. The same goes for syntagmatic conditions. When a variable is specified for the parameter *open* in the DSL file, there will be an extra entry in the list of values reading 'z. other', as in (17). When, in such a case, the user types a 'z' then any value may be entered for the variable. It will be automatically added to the list of values for that variable in a dynamic way. This dynamism implies that, as long as there is at least one language in the data base that has the value concerned for that variable, it will be presented on the screen with the predefined values for any language to be coded after. If it is removed from the data base, it will disappear from the menu presentation altogether. In order to attain consistency in the terms and concepts used in a certain domain, an alphabetical list of abbreviations and definitions may be compiled on a file by the user (this is the ABBREV file in figure 2). It may be consulted and browsed through at any stage during the execution of the program.

In presenting the successive variables on the screen, prompting the user for values, the program is sensitive to the paradigmatic conditions specified in the DSL file. This means that series of variables may be excluded from being presented on the screen at all. In such cases, the program will automatically assign a 'not relevant' value to these variables. Whenever a variable is presented on the screen, the user always has to provide a value. If it is not known, or it is not in the set of predefined values, the user may type a question mark (?), which is the system's built in missing value. The question mark may always be replaced at a later stage by a 'real' value.

When finished with some variable, the complete value set is added to the domain specific data base file for the language under consideration. One can then proceed to the next variable.

At any stage, the user may interrupt the entering of data, or shift to another language. The program will automatically continue with the variable next in the hierarchy on the ensuing occasion.

Data already entered may be inspected by choosing the 5. option in the menu in (16) and also modified by choosing the 4. option. If a modification, as a consequence of paradigmatic conditions, has implications for data already provided for lower variables for the language concerned, the program will notify the user of this. The data base should then be reorganized in the necessary way.

The end product of the application of the QTP data entry program is a QTP data file where all the variables with the specified values, syntagmatic conditions and scales for all the languages are stored. It is this QTP data file which serves as the input to the statistical and analytical computer programs.

4.2 TRANS: data transformation

TRANS is an interface module which has two functions. The first is the integration of the data stemming from the QTP program into a centralized data base the format of which is the same as that of the QTP files. This data base may contain data stemming from other domains and sources. Since all the domain specific QTP files are assumed to be structured in the same way, the TRANS interface provides for the possibility of inter-domain retrieval and analysis.

The second function of the TRANS interface is to transform the QTP data into a format readable by SPSS, one of the most widely available statistical packages. It recodes the alfanumeric value labels of the QTP data into the numeric ones preferred by SPSS, and organizes the data into a rectangular data matrix with the languages as the cases. In this transformation, part of the advantages of the DSL language are, of course, lost, such as the multi-value option, and the separation between values, scales and conditions. For that reason, some ad hoc provisions have been built into the TRANS program. For example, scale values may be optionally appended to the variable values proper, giving complex value types such as 'AdjN_obl' or 'NAdj_opt'. Syntagmatic conditions, however, cannot as yet be handled at all. Multi-value variables may be treated in two ways. By default, they are split up in as many SPSS variables as is necessary for the highest actual number of values to be represented. For example, when a variable Vn was defined as 'multi=5' in the QST file, and actually has 4 values as a maximum for some language, there will be four variables Vn_A through Vn_D in the SPSS file. A language that has only 2 values for Vn will have missing values assigned to its Vn_C and Vn_D. If splitting the values over a set of sub-variables is thought to be undesirable, another possibility is to concatenate them, as in the case of scale values. In order to maintain consistency, the values are concatenated in alphabetical order. The effect of this value transformation is that it changes the variable concerned from coding a mixed type to coding a type. Under this option, scale values are suppressed.

For ease of use, TRANS provides a complete SPSS/PC setup, and adds the original variable names and values as labels; this greatly improves the readability of the SPSS output. Appendix D gives an example of such a SPSS set up file.

4.3 The DBL linguistic data base

For querying and related operations a separate program - Data Base on Languages (DBL) - has been developed. This program has been constructed around two sets of data. The first is the file that results from the integration of several domain specific QTP files, i.e. the just discussed centralized data base brought about by the TRANS program. The second is a computerized version of Ruhlen's (1987) language classification, represented in the form of a genetic tree with 18 major phyla, 1825 intermediate sub-phyla and groups, and 5273 languages for their terminal nodes [6]. These two sets of data are connected into a combined data base. Using the DBL program, this combined data base may be explored in two ways: by searching and by sampling. For the sampling option we refer the reader to Rijkhoff et al. (1992). The querying will be discussed directly below.

4.3.1 Querying the DBL data ase

Searches through the data base may be made via the genetic classification as reflected in Ruhlen or via the language data as such. Using the genetic approach one may interactively browse through the genetic trees picking out names of phyla and subphyla. This may provide the user with the languages in their genetic relationships to other languages and language groups, typically leading to the type of screen display given in figure 3 below. Per node in the tree the data that are available with respect to the dependent languages may be retrieved. Figure 4 gives a (simplified) example of such a survey.

```
                    BABEL

Caucasian                              Indo_Hittite

North                                  Indo_European

Northwest                              Germanic

Abxaz-Abaza                            West

                                       Continental

                                       West

*Abxaz*                                *Dutch*

SOV/free                               SVO/V2
post                                   prefprep
NAdj/AdjN                              AdjN
GN                                     GN
NumN                                   NumN
RelN                                   NRel
```

figure 3

```
Group: Germanic
Number of sister nodes: 8
Descending nodes: 26
Descending languages: 16
Information on: 7

SVO=3 / SVO/V2=2 / V2=1 / SOV=1
prep=6 / prefprep=1
AdjN=7
GN=7 / NG=5
NumN=7
NRel=7 / RelN=1
```

figure 4

Adopting the language data approach as represented in the QTP file one can
make queries consisting of boolean expressions over variables and values. In
these expressions, the same conventions are used as for the representation of
syntagmatic conditions in DSL file. An example of a simple query is given in
(18); a more complicated one is given in (19).

```
(18)  (V0.1=SOV OR V0.1=SOV/free)  AND  V0.2=post

      abx bas chu dar did geo kom lez

(19)  (((V0.1=SOV OR V0.1=SOV/free) AND V0.2=post) OR
        ((V0.1=VSO AND V0.2=prep))

      abx bas bre chu dar did geo kom lez wel
```

Query (18) selects the languages with basic SOV and SOV/free word order that
have postpositions. (19) selects these languages and those that have basic VSO
order and prepositions.
It is important to note that scales and conditions may not be included in
queries. They may, however, be reproduced in displays.

4.4 The LINFER inferencing program

LINFER is the fourth module of our scheme. It has been developed specifically
for the analysis of analytical linguistic data. As mentioned earlier, the host
of computer programs that are commercially available for the analysis of data
organized according to a scheme of variables and values, such as SPSS, SAS or
BMDP, just to name a few, do not provide all types of analysis procedures that
work in language typology calls for. They do contain facilities for both data
description (cf. frequency counts and crosstabulations) and analysis (cf.
techniques such as Cluster Analysis, see below) that may be of great help when
trying to come to grips with the host of secondary linguistic data resulting
from a questionnaire. However, in order to apply these techniques, especially
the more sophisticated ones, one often has to resort to recoding, reorganizing
or reinterpretation of the data. Furthermore, as has already been observed
above, many statistical techniques presuppose data types that are of a higher
order than the nominal type of data that is usually employed in the field of
language typology. By contrast, the LINFER program that we offer directly
incorporates several of the methods and quantities that have been introduced
into the field of language typology by Greenberg and subsequent scholars. The
most important of these will be presented below.

4.4.1 Simple implications

Probably the most well-known type of rule in the field of language typolog
is the implication; deriving implications is the basic step in the inferenc
process built into the LINFER program. In the first pass of this process, th
user may select a set of variables from the QTP type data base. The LINFE
program will derive any simple implications existing between the selecte
variables. One may select any sub-set of the variables in the data base
including the whole set. The only variables that cannot be selected are thos
for which only one value is relevant for the languages in the sample. Th
program will automatically disqualify such variables.
In the context of the program, implications take the following form:

(20) V1=val_1 -> V2=val_2 , < Fr=x, Fa=y, Fc=z, Pv=p, Pw=q >
where:

- V1 and V2 are variables from the subset chosen;
- val_1 and val_2 are values for V1 and V2 respectively;
- x, y, z, p and q are values between 0.0 and 1.0;
- Fr is the fraction of languages in the sample for which the implication is relevant, i.e. that have value val_1 for variable V1;
- Fa is the fraction of the relevant languages for which the implication applies;
- Fc stands for what we call coverage, i.e. the fraction of languages for which the conclusion is relevant, but not the premise;
- Pv is the chance that a language has V2=val_2, given the number of different values for V2;
- Pw is the chance that a language has V2=val_2, given the overall distribution of values for V2

The meaning of the F and P rates can be more easily appreciated on the basis of a concrete example. Let us assume that in a data base with data on 30 languages there are 20 languages that have V1=val_1. Of these 20 languages 18 also have V2=val_2. In addition there is another language (not one of the 20) which also has V2=val_2 but it does not have V1=val_1. Given the above, the F rates will be as follows: Fr=0.667 (=20/30), Fa=0.9 (=18/20), and Fc=0.947 (=18/19), respectively. In case there are 5 different (non-missing) values for V2 for the languages in the data base, Pv will have value 0.2 (=1/5). If all 30 languages have a value for V2, then Pw - i.e. the real chance on a language having V2=val_2, is 0.62 (=18/29).

By means of the F and P rates we thus quickly learn how potentially interesting the particular implication is [7]. Either of the phenomena covered by the implication may be widespread or they may actually occur only in a handful of languages. Any implication that has Fa=1.0 is an absolute universal [8]. Though an absolute universal is of interest irrespective of how many languages actually display it, it may be desirable to know to what extent the two phenomena in question actually do co-occur, i.e. how often the phenomenon referred to by the conclusion is found without that denoted by the premise. Consider, for instance, the well known simple implication in (21).

(21) V28.3=Gender -> V28.3=Number n=24

<Fr=0.55, Fa=1.00, Fc=0.73, Pv=0.25, Pw=0.87> Absolute

lgs: alb bul bye cat clg cze dan dut frs ita lit mac mal
nor pol rus scr slva slve spa swe ukr uso wel

The implication in (21) - in words: if the demonstrative agrees with the noun in gender, then it also agrees with the noun in number - holds for 24 of the 44 languages currently in the word order data base, i.e. for over just half of the languages (Fr=0.55). There are no languages which display gender agreement of demonstrative and noun and which do not simultaneously exhibit number agreement between the demonstrative and the noun. Consequently, the implication holds for all the languages manifesting the premise. There are, nonetheless, 7 languages in which the demonstrative agrees with noun in number, but not in gender, i.e. languages that have the conclusion, but not the premise. This leads to a Fc value of 24/31, or 0.73.
Pv is 0.25 (i.e. 1/4) since there are 4 different non-missing values for variable V28.3 (noDemAgr, Case, Gender and Number); Pw is 0.87 (i.e. 32/37)

since 33 out of 38 languages with non-missing values for V28.3 have value Number.

The procedure that computes implications is parametrized to the extent that it will only retain the implications with F values above a certain user defined minimum. Thus we may be interested only in implications that hold for all the languages which exhibit the premise and the conclusion (minimum Fa is 1.0). Or we may be more liberal and allow for implications that hold for only 90% or 75% of the languages for which the premise and conclusion are relevant. Or we require that any implication should be relevant for at least 1 in 4 languages (Fr=0.25).

The implications computed by the LINFER program may be ordered according to several criteria: in terms of any of the three F values, where the implications with the highest of the F value selected will be placed on top of the list and implications with successively lower F values will follow, or in terms of either the premise or conclusion of the implication.

Often, especially with large sets of variables, and relatively low minima for the Fa and Fr values, quite large numbers of implications may result from this procedure, say several hundreds or even thousands, the vast majority of which may be without much interest. Therefore, we need to be able to restrict the sets. Apart from tuning the F values, this can be achieved in several ways. The first method is via the reduction option. When this option is activated, for any pair of simple implications that have the same factors, only the one with the highest F rates will be retained. For example, in a set of implications computed by the program we may have:

```
(22)  a. V1=val_1  ->  V2=val_2   < Fr=r1, Fa=a1, Fc=c1 >
      b. V2=val_2  ->  V1=val_1   < Fr=r2, Fa=a2, Fc=c2 >
```

With the reduction option on, if Fa is the main sorting item, only the implication with the highest Fa value is retained. (In case a1 and a2 are equal: the one with the highest Fr value is retained). When all F values are equal, both are retained.

A second way of output reduction is the clustering of variables into groups. Say we are interested in the relationship between the order of the modifiers of the noun relative to the noun or relative to each other and the existence of or type of NP internal agreement. We can then select the variables pertaining to modifier noun order, and assign them to group 1. The variables reflecting agreement properties within the NP can then be assigned to group 2. Thanks to the clustering option, the LINFER program will compute only the implications holding between the group 1 and group 2 variables. It will ignore any implications obtaining within the respective groups. So, if we have some variable of group 1 as a premise, we may have only variables of group 2 for a conclusion, and the other way round.
The group option is very useful, since it enables one to find lower level implications, i.e. holding for smaller sets of languages which would otherwise be missed or obscured.

Yet another reduction of implications can be achieved by assigning levels to groups. With this option the program will only derive implications of which the variable in the premise has a lower level than that in the conclusion. Implications in language typology are generally perceived as neutral towards inductive or deductive reasoning over sets of variables. By assigning levels to (sets of) variables, and allowing higher level variables only as a conclusion, an inductive perspective is created, in which the higher level variables might explain, or at least could be said to be necessary for, the lower level ones. In an alternative mode, this scheme may be reversed, and the perspective made deductive rather than inductive. If we would, for example, assign level 1 to a variable AdjNorder (= adjective-noun order), and level 2 to a variable OVorder (= object-verb order), then, over a certain data base, we might derive the following implications under the induction and deduction options respectively:

```
(23)  a. (inductive)          AdjNorder=AdjN  ->  OVorder=OV
      b. (deductive)          OVorder=OV  ->  AdjNorder=AdjN
```

Although, in a technical sense, the occurrence of these implications under the respective options do not necessarily confirm the level status assigned to the variables, this option helps to create a certain experimental perspective on the data. For example, if the level ordering were to be right in the above cases, one would expect Fa and Fc values of (close to) 1.0 for deductions, but not necessarily for inductions. This point will be further elaborated in section 4.4.2. As a side effect of the level option the amount of implications generated is further reduced.

If in the body of derived implications, we have a tuple such as that in (22) above, and both are absolute (i.e. Fa=1.0), then we have an equivalence, that will be represented by the program as follows:

```
(24)  V1=val_1  <->  V2=val_2  ,  n=lgs

      < Fr=r, Fa=1.0, Fc=1.0 >  Equiv
```

where r=r1=r2. One of the equivalences that was found in the word order data is given in (25):

```
(25)  V28.1=Gender <-> V28.3=Gender        n=24

      <Fr=0.55, Fa=1.00, Fc=1.00, Pv=0.25, Pw=0.62> Equiv

      lgs:  alb bul bye cat clg cze dan dut frs ita lit mac mal
            nor pol rus scr slva slve spa swe ukr uso wel
```

(25) states that among the languages in the data base, whenever a language displays gender agreement between adjective and noun, then there is gender agreement between demonstrative and noun, and vice versa.
In the light of the options described below, both versions of an equivalence are retained, even under the reduction option, as are all the implications with equal F rates.

On the basis of a set of simple implications compiled over some selection of variables in a data base, other - secondary and tertiary - quantities may be computed. It is to these that we now turn.

4.4.2 Complex implications

In order to give absolute validity to linguistic implications, Hawkins (1983: 75f) combines simple implications into complex ones. One of the most frequently cited of Hawkins' complex implications is the Preposition Noun Modifier Hierarchy presented in (26).

```
(26)  Prep -> ((NDem OR NNum -> NA) AND
                (NA -> NGen) AND
                (NGen -> NRel))
```

In essence, the complex implication in (26) states the following: if a language is prepositional, then, if demonstratives or numerals are placed after the noun, then the adjective, genitive and relative clause are also to the right of the noun. Or: in prepositional languages, the longer a nominal modifier is, the easier it moves to the right of the nominal head.
An option has been built into the LINFER program to derive this type of complex implications. There are three versions of this option: conjunctions; disjunctions; and full logical inference. They will be treated in some detail below. Since the complex implications are derived from simple implications already computed in the first phase, only those implications are considered that surpassed the minimum values for the respective parameters. If one wants all implications to be found in the data to be considered as elements of complex ones, no constraints should be initially set on their derivation. This is particularly important for disjunctions since several low scoring implications may lead to high scoring disjunctions.

4.4.2.1 Conjunctions

Under this option, simple implications are combined by way of the logical AND operator. Assuming that in our body of derived simple implications we have the three implications in (27) the program will derive the chain in (28):

```
(27)  V1=v_1 -> V2=v_2
      V2=v_2 -> V3=v_3
      V3=v_3 -> V4=v_4

(28)  V1=v_1 -> ( V2=v_2 -> ( V3=v_3 -> V4=v_4 ) ), < Fr, Fa >
```

In (28), Fr is determined by the conjunction of the relevance sets for the premises of (27); Fa is the subset of this to which V4=v_4 applies [9]. Some concrete examples of such a chain of implications from the word order data base are given below. First a complex implication based on two simple ones.

```
(29)  V3_0=case -> (V29_6=case -> V29_4=case)    n=22

      < Fr=0.500, Fa=1.000 >

      lgs:  alb bye clg cze dar did est fin geo hng lez lit
            mar mor pol rus scr slva slve udm ukr uso
```

This (exceptionless) complex implication states that if a language codes recipients in ditransitive clauses by means of inflectional case, as opposed to adposition or word order, then, if it also has case marking on objects then it has case marking on subjects. The following chain is derived from four simple implications:

```
(30)  V15_3=P1 -> (V4_3=all -> (V27_1=S ->
                         (V34_1=yes -> V34_3=no)))    n=12

      < Fr=0.295, Fa=1.000 >

      lgs: bye clg cze est lit mac pol rus scr slva slve ukr
```

This complex implication should be read as follows: if a language requires all Wh-words in multiple questions to be fronted, then, if all basic orders are possible in declaratives without left or right dislocation, and if there is subject agreement on the verb, then if the language may drop the pronominal subject, it has no expletive. This implication is in force for 12 languages in the word order data base; there are no exceptions.

In the program, a lower limit may be set to the relevance and applicability values of such chains independently from those that were determined for the simple chains. The program will compile all chains of whatever length. This option may bring about more complex relations between sets of variables, and may put one on the track of cooperating and conspiring forces in a grammar, such as the relativization and promotion conspiracy discussed, for example in (Croft 1990), especially if these are based on more than two factors.

In relatively rich domains, a great number of chains may be compiled, especially when no levels are assigned to the variables. With the assignment of levels, complex hypotheses on hierarchies between (sets of) variables may be tested. The user may tune the minimum F values in order to get the number of resulting rules considered to be both interesting and manageable.

4.4.2.2 Alternative factors

Within a body of simple implications, a number of implications may be found that share either the premise or the conclusion. The LINFER program has a built in option to collect all implications that have such a common factor. So, if among the set of implications for some language sample, the following are to be found:

```
(31)  V1=val_1 -> V2=val_2
      V1=val_1 -> V3=val_3
      V1=val_1 -> V4=val_4
```

the program will derive

```
(32)  V1=val_1 -> V2=val_2 OR V3=val_3 OR V4=val_4

      < overlap = x >
```

(32) may be interpreted either deductively as: 'val_1 is expressed in languages by one or more of val_2, val_3 or val_4', or inductively as: 'if a language has val_1, then it is either a val_2, val_3 or val_4 language'. An important aspect here is the amount of overlap between the languages to which any of the subfactors in the conclusion applies, expressed by the overlap factor x. This is a value between 0.0 and 1.0. Overlap equal to 0.0 means that the OR operators are, in fact, exclusive: no two subfactors are shared by any language. Overlap equal to 1.0 means that the complex factor may be read as a conjunction of subfactors, in other words that we are dealing with an AND chain.

The above type of implication may gain in interest if the factors in the conclusion stem from the same level. This may be brought about by using the group and level options. A special situation arises when we find only one variable in the conclusion. When this is combined with overlap = 0.0, we may be dealing with a subtypology of the languages of the type characterized by the premise. When the overlap equals 1.0, and all values of the variable are in fact present, we may have a case of dominance (see also section 4.4.5). Depending on the perspective one wants to adopt, the resulting rule sets may be ordered according to an increasing or decreasing overlap value. It is possible to determine a threshold value for overlap, a maximum in case of increasing order and a minimum for decreasing order.

This scheme may be reversed to the extent that the program may be instructed to collect implications that have the same conclusion instead of the same premise. So, if the following simple implications were derived:

```
(33)  V2=val_2 -> V1=val_1
      V3=val_3 -> V1=val_1
      V4=val_4 -> V1=val_1
```

the program can combine them into:

```
(34) V2=val_2 OR V3=val_3 OR V4=val_4 -> V1=val_1

      < overlap = x >
```

Using the deduction/induction option, the desired perspective may be created just as in the case of simple implications.

Since the alternative factors option merges subsets of languages, it is

particularly suited for raising the F values, Fr values in the case of combined premises, and Fa values for combined conclusions. Threshold values for these may again be determined independently.

4.4.2.3 Full logical inference

A third option is the derivation of all complex implications, including combinations of AND and OR operators, such as those in (26). The algorithm generates all possible expressions of propositional logic in a systematic short-to-long manner, and fits all existing variable-value combinations in each scheme. Of each fully specified expression the F values are determined. If they are above the predetermined threshold values, the implication is retained for later inspection. Optionally, the NOT operator may be introduced into the scheme, but only on the lowest level, i.e. as an alternative for the = operator in variable-value pairs [10].

An option restricts the length and complexity of the generated strings. And yet another allows for the application of the group and level options. Implication (26) above came out of this process in the following fashion:

```
(26') ((V0_2=prep OR V0_2=prefprep) AND
       (V20_9=NDem OR V20_11=NNum) AND
       V20_15=NAdj AND V21_2=NG)          ->  V20_18=NRel   n=4

      < Fr=0.091, Fa=1.0 >

      lgs: clg mal pol rus
```

This confirms (26) for the sample in the word order data base.

4.4.3 Explaining exceptions

If some implication has Fa < 1.0, then it does not hold for all the languages that exhibit the premise, in other words it has exceptions. The program lists the languages that constitute an exception to a given implication, provided that their number is less than that of the languages to which the implication does apply (i.e. there is a Fa value of at least 0.5). With the explanation option on, the program will search for factors that could possibly explain the distinct behaviour of the languages that constitute the exception to the implication. Let us assume that: Sa is the set of languages to which some implication In applies, Se is the set of languages that are an exception to In, and Le is a language in Se. Given the above, the program will trace all the variables and factors that are relevant for all the languages in Sa, but not for Le. These are reproduced as being irrelevant for Le only. In addition the program will trace all the variables and factors that are relevant for Le, but not for any language in Sa. These are reproduced as being relevant for Le only. An example from the word order data base is (35):

```
(35)  V20_15=AdjN -> V20_9=DemN      n=42

      < Fr=1.00, Fa=0.96, Fc=1.00, Pv=0.50, Pw=0.98 >

      lgs:  abx alb bre bul bye cat clg cze dan dar did dut
            eng est fin fre frs geo ger gre hng ita kom lez
            lit mac mal mar mor nsa nor pol rus scr slva
            slve spa swe udm ukr uso wel

      exc:  bas chu
      Basque              V20_9=NDem
      Chuvash             V20_9=?

      ** Possible explaning factors: **
      Basque        V20_17=IntNAdj
      Basque        V20_20=RelGenNum
      Basque        V20_21=AdjArt/Dem
```

Basque and Chuvash are the only exceptions to the rule that if a language preposes the adjective then it preposes the demonstrative. For Chuvash the crucial information is not (yet) available; for Basque the opposite noun-demonstrative order applies. The data base is now searched for variable-value pairs that are unique for Basque. Three of these are printed here. Basque is the only language in the data base that has the noun in between the intensifier and the adjective; that obligatorily preposes full relative clauses, genitives and number in that order; and that has the article and the demonstrative at the end of the noun phrase. Whether these - and other factors, not printed here - constitute an explanation for the exception is, of course, a matter of further interpretation and research.

4.4.4 Parameters

In linguistic theory, variables that have a relatively high level of explanatory power are sometimes called the parameters of the theory. The LINFER program has an option that determines the possibly parametric variables to be found in the set of variables in a QTP type data base. A variable is considered to be parametric if and only if it has one, and not more than one real (i.e. non-missing) value for all languages in the data base. The set of thus established parametric variables is ordered according to the number of times it is found as a factor of a simple or complex implication computed over the data base. If the inference scheme is set to neutral, any occurrence as a factor is counted as 1. In the inductive or deductive mode, an occurrence is weighted according to its position in an implication. For an induction, an occurrence is assigned more weight the further it occurs to the right (e.g. the premise of a simple implication counts for 1; its conclusion for 2; the final conclusion of a implication such as (30) counts for 5); for a deduction the highest weight is assigned to the occurrence furthest to the left. The total sum determines the parametric weight of some variable. If groups are assigned, the program will compute a level for any group, based on the sum of the parameter weights for the group variables. Some of the parameters found for the word order data base are:

```
(36)  V0_1  (greenbergian classification)
      V3_4  (dative shift)
      V14_1 (Q-particle in yes/no questions)
```

4.4.5 Dominance

In (Greenberg 1963), a variable-value combination is called dominant relative to some variable if it occurs with all values of that variable. An example is to be found in the following table, which gives the co-occurrences of noun-adjective and noun-demonstrative orders in Greenberg's sample:

	NA	AN
DemN	12	7
NDem	11	0

DemN is dominant because it occurs with all values of the adjective-noun variable, while NDem does not (which is therefore recessive). For the same reason NA is dominant and AN is recessive. In the program, the notion of dominance is generalized to: Var1=val_1 is dominant over variable Var2 i Var1=val_1 occurs with all (non-missing) values of Var2 as the only value fo variable Var1. Var1 is called the controlling variable and Var2 a controlle variable of the dominance relation. In order to stay close to the origina concept, by default only non-multi valued variables are considered as th controlling variable for dominance. In these cases we will use the term pur dominance. Optionally, multi-valued variables may be taken as controllers Such relations will be called mixed dominance. In the resulting list variable-value combinations are ordered according to the number of dominanc relations they maintain. An example of pure dominance and mixed dominance i the word order data base are given under (37a) and (37b) respectively:

```
(37) a. Variable V27_3  (subject person agreement on the verb)

        Value "123" has pure dominance over all values of:

            V27_6 (obj agr verb)
            V27_7 (obj gender agr verb)
            V27_8 (obj person agr verb)
            V27_9 (obj number agr verb)
            V27_10 (form obj agr marker)
            V28_2 (agr pred adj subj)
            V28_3 (agr dem noun)
            V28_4 (agr art noun)
            V28_5 (agr numr noun)
```

b. Variable V29_1 (type of bondness)

Value "suff" has mixed dominance over all values of:

```
V27_6 (obj agr verb)
V27_7 (obj gender agr verb)
V27_8 (obj person agr verb)
V27_9 (obj number agr verb)
V27_10 (form obj agr marker)
V28_3 (agr dem noun)
V28_5 (agr numr noun)
```

Apart from being a quantity in their own right, dominance relations may lead to the derivations of markedness patterns. Since the dominant value of some variable implies the greatest diversity vis a vis the controlled variables, it qualifies for the status of unmarked value (cf. (Croft 1990) on features of markedness).

4.4.6 Typological hierarchy

A variable has strict typological value ordering if its values may be ordered according to the following scheme:

(38) val1 < val2 < val3 < ... < valn

where value valj is never relevant for some language without valj-1 being relevant for that same language [11]. The variable concerned does not have to be relevant for all languages. A well-known example of such a variable in the typological literature is Number, that has the following ordering:

(39) Number = singular < plural < dual < trial

The LINFER program has a built in option to trace variables with typological ordered values. In the word order data base, the following are found, among others (again, this concerns of course the 44 European languages in the data base only):

```
(40) a.    NumN    < NNum
           lgs=43  < lgs=5

     b.    OrdN    < NOrd
           lgs=42  < lgs=2

     c.    AdjN    < NAdj
           lgs=44  < lgs=21

     d.    suff    < pref
           lgs=40  < lgs=4
```

Partial typological ordering is a relaxation of strict ordering. A well-known example is the colour hierarchy as established in (Berlin & Kay, 1969). In general, variables with partial ordering must adhere to the following scheme:

(41) valueset1 < valueset2 < valueset3 < ... < valuesetn

where valuesetj is a non-empty subset of values of the variable concerned. In order for the relation to hold, any language should choose its values such that never a value is selected from subset valuesetj if there is a subset in the range valueset1 to valuesetj-1 that it has not selected a value from. Examples from the word order domain:

(42) a. (expression definiteness)
art, dem, posspro < word_order < ClDoub

b. (object agreement verb)
pat < rec

c. (passive subject)
pat < rec, ben < AdpObj

(42c) gives support for the Semantic Function Hierarchy as to be found in Functional Grammar (Dik 1989).

4.4.7 Language clustering

A last step in the analysis is the compilation of clusters of languages. For the clustering of cases in a data matrix, several methods have been developed, all going under the label of cluster analysis (cf. SPSS/PC Advanced Statistics). For all these methods, clustering takes place over variables that are at least of the ratio level: the distance between cases is determined on the basis of the amount of difference between the values for the clustering variables. Not many analytic linguistic variables will qualify for such an analysis. Still, clustering, especially the hierarchical version, appears to be an attractive way of representing the likeness of languages along certain specific dimensions. By choosing the right variables or parameters, we may establish hierarchies of typologically related languages. We therefore included two options. The first is a straightforward built in strategy that groups languages into clusters of a predetermined size between 2 and n-1 where n is the number of languages in the data base. These clusters are determined on the basis of the number of implications that the languages in the cluster share. The strongest cluster of size s is the one whose s languages share the greatest number of implications. As an example, we give some of the strongest clusters considering the 100 strongest implications in the word order data base, for cluster sizes 3 and 4, respectively:

(43)

```
LANGUAGE CLUSTERS OF SIZE 3:
Cluster 1    lgs: bul bye ger
Cluster 2    lgs: bul bye gre
Cluster 3    lgs: bul slva ukr
Cluster 4    lgs: bul fin lit
Cluster 5    lgs: bul bye lit
Cluster 6    lgs: bul bye mac
Cluster 7    lgs: bul fin mac
Cluster 8    lgs: bul bye pol
Cluster 9    lgs: bye ger lit
Cluster 10   lgs: bul bye scr
```

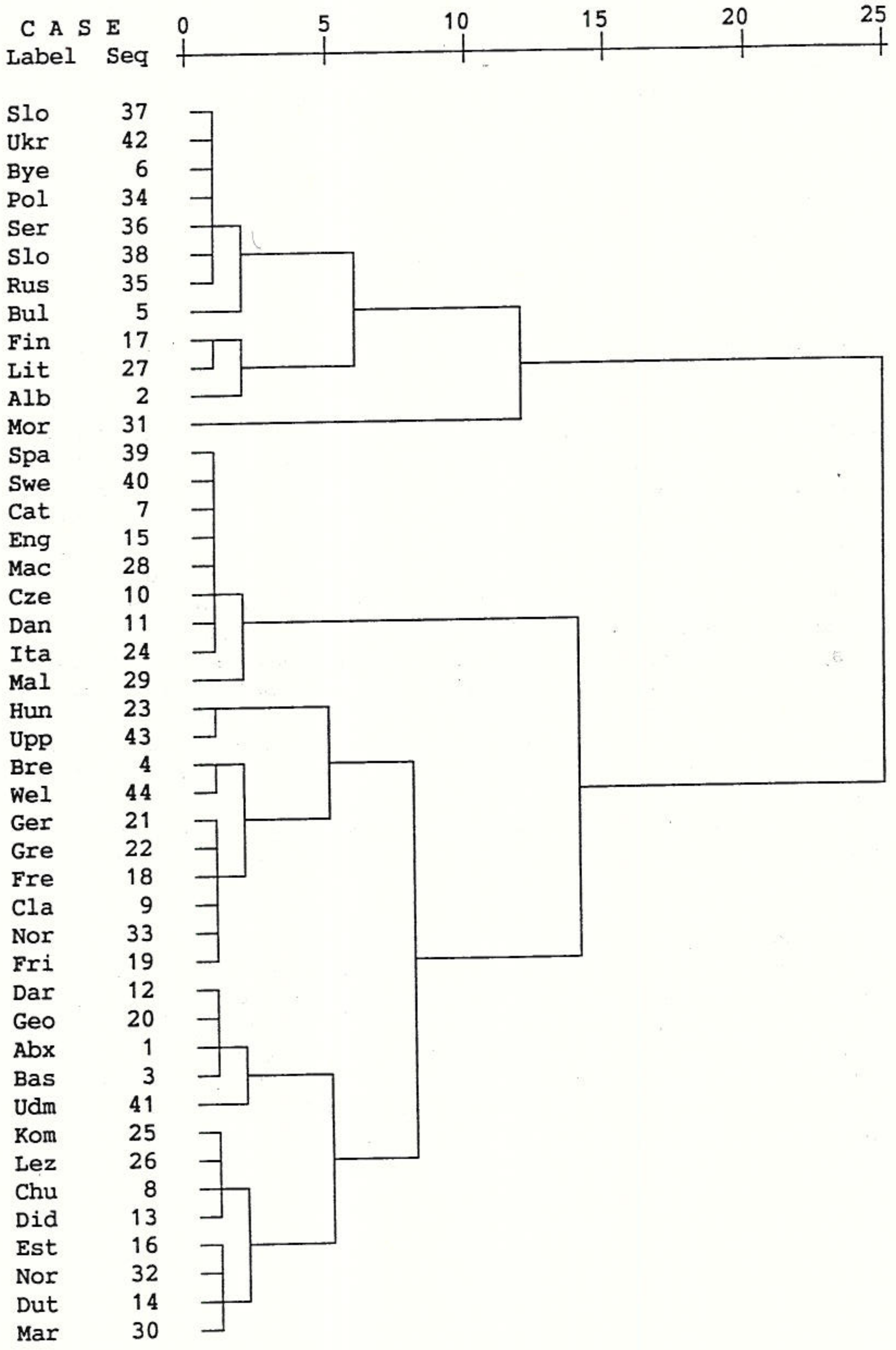
(44)

```
LANGUAGE CLUSTERS OF SIZE 4:
Cluster 1   lgs: bul bye fin lit
Cluster 2   lgs: bul bye ger mac
Cluster 3   lgs: bul bye fin mac
Cluster 4   lgs: bul bye ger pol
Cluster 5   lgs: bul cze mac pol
Cluster 6   lgs: bul bye ger scr
Cluster 7   lgs: bul bye ger slva
Cluster 8   lgs: bul bye ger slve
Cluster 9   lgs: bul bye ger ukr
Cluster 10  lgs: bul cze fin ger
```

In general, a great many clusters will result from this procedure. The program may be instructed to retain only a reduced number of the strongest clusters.

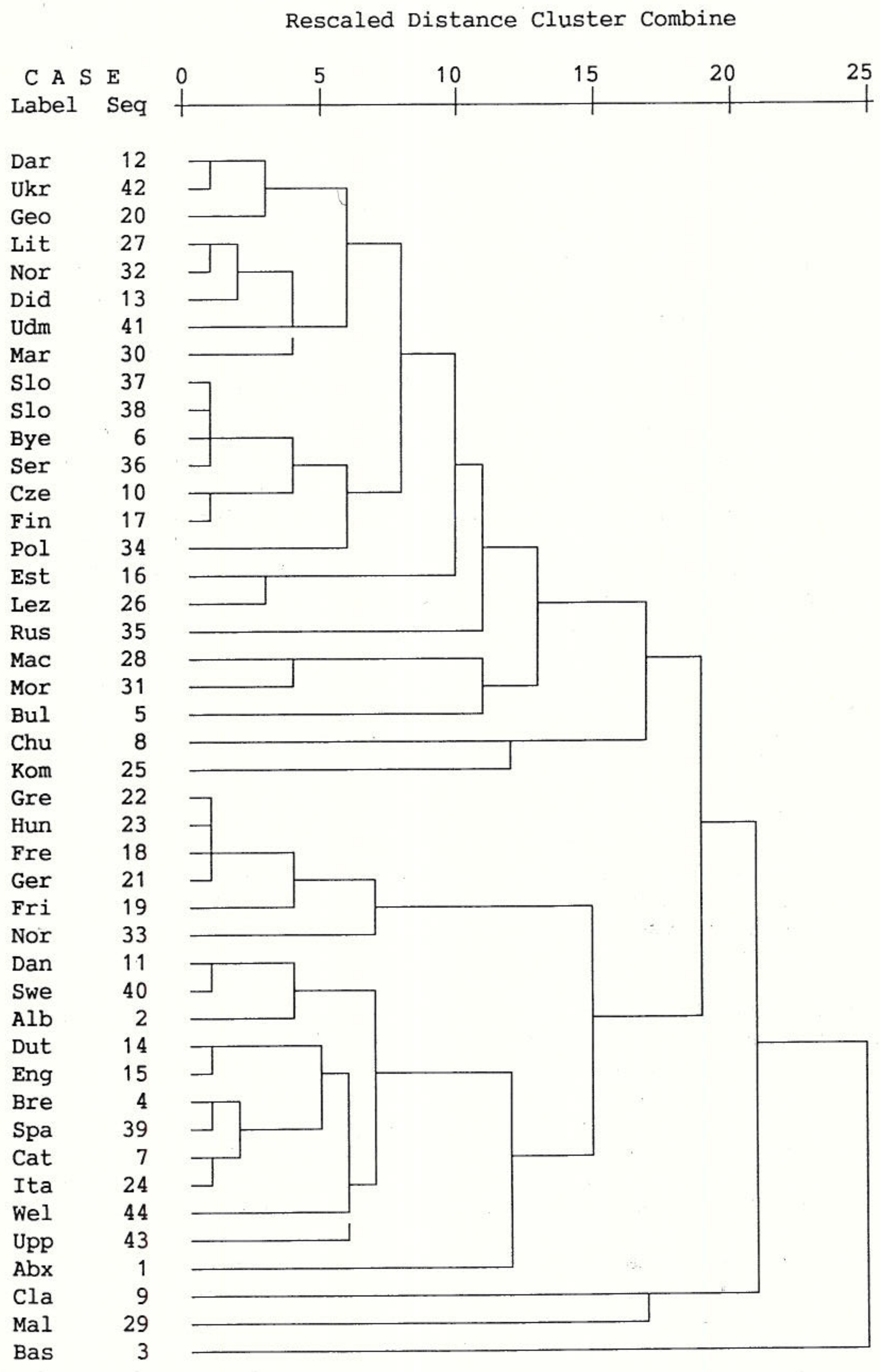
This clustering technique is quite straightforward, and of limited interest. Many standard statistical packages have built in cluster algorithms of a relatively high level of sophistication that can provide the researcher with more interesting clustering results. Since the secondary linguistic data do not normally qualify for such analyses in a direct way, tertiary variables may be derived through the LINFER module that do have the right properties. For example, let I1...In be the set of simple implications derived for the languages in the data base. Now for any language L in the data base a set of variables V1...Vn is generated such that Vj has value 1 if Ij applies to L and 0 if it does not apply. The data matrix created in this way is written to a file in the right format and extended with the necessary descriptive information that makes it accessible for the statistical package SPSS; SPSS disposes of a series of cluster analytical procedures that may be run on this derived data matrix [12]. Instead of value 1, LINFER will optionally write either the Fa value or (1-Fr) for each implication. With Fa, most weight will be given to the implications with the heighest application rate. With (1-Fr), the implication will be given more weight by the clustering procedure if it is relevant for less languages, i.e. if the collocation of the languages is more specific.

Running SPSS Cluster analysis on a data matrix for the 44 languages based on the 100 implications that had the highest Fa values over the complete set of word order variables gave the diagram of figure 5. One over the variables describing word order in the noun phrase gave the diagram of figure 6.
It should be stressed that this diagram is not the reconstruction of a genetic tree, but a graphic representation of the level of closeness of languages as far as their cooccurence in implications over the word order variables is concerned. Such representations may, however, be used in constituting typologogies.



C A S E
Label Seq
0
5
10
15
20
25
Slo 37
Ukr 42
Bye 6
Pol 34
Ser 36
Slo 38
Rus 35
Bul 5
Fin 17
Lit 27
Alb 2
Mor 31
Spa 39
Swe 40
Cat 7
Eng 15
Mac 28
Cze 10
Dan 11
Ita 24
Mal 29
Hun 23
Upp 43
Bre 4
Wel 44
Ger 21
Gre 22
Fre 18
Cla 9
Nor 33
Fri 19
Dar 12
Geo 20
Abx 1
Bas 3
Udm 41
Kom 25
Lez 26
Chu 8
Did 13
Est 16
Nor 32
Dut 14
Mar 30

figure 5



Rescaled Distance Cluster Combine
C A S E
Label Seq
0
5
10
15
20
25
Dar 12
Ukr 42
Geo 20
Lit 27
Nor 32
Did 13
Udm 41
Mar 30
Slo 37
Slo 38
Bye 6
Ser 36
Cze 10
Fin 17
Pol 34
Est 16
Lez 26
Rus 35
Mac 28
Mor 31
Bul 5
Chu 8
Kom 25
Gre 22
Hun 23
Fre 18
Ger 21
Fri 19
Nor 33
Dan 11
Swe 40
Alb 2
Dut 14
Eng 15
Bre 4
Spa 39
Cat 7
Ita 24
Wel 44
Upp 43
Abx 1
Cla 9
Mal 29
Bas 3

figure 6

4.4.8 Further issues

A number of extra options and aspects of the LINFER program have remained outside the discussion so far. A few of them will be presented briefly below.

4.4.8.1 Changing values

In section 3 it was mentioned that complex values may be subjected to analysis via a formal procedure. The LINFER program can do this in two ways: by recoding and by rewriting. In the case of recoding, a value is substituted by another value. A case in point is illustrated in (45).

```
(45)  a.    VO = ( SVO, VSO, VOS )
      b.    VO = ( SVO, VSO, VOS ) ,
                  ( Basic_Order, Transitive_Orders )
```

(45a) shows the substitution of the three values in the list by the value to the left of the equals sign for all variables. In (45b) the substitution effects only the two variables specified in the second list.
Another way of value transformation is by rewriting. This mode takes advantage of the analyzability of the variable values mentioned above. The rule in (46) will rewrite to VO all values that contain V and O as elements, in that order. The Kleene star, that forms part of the meta-language, has the conventional meaning of standing for zero or more symbols:

```
(46)  *V*O* -> VO
```

Rewriting may also be restricted to a subset of the variables.

4.4.8.2 Scales

When a scale is in force for some selected variable, it may play a role i implications in two ways. First, the scale value may be used to distinguis between factors. When it is used to do so, it is added to the factor, so w may then have the following implications:

```
(47)  a.  V1=val_1 (obligatory)  ->  V2=val_2a
      b.  V1=val_1 (optional)    ->  V2=val_2b
      c.  V3=val_3a  ->  V4=val_4a (formal)
      d.  V3=val_3b  ->  V4=val_4b (informal)
```

This option may introduce implications that do not emerge without the sca distinctions due to the fact that they remain under any reasonable lower lim of Fa or Fc values. The second way of using scales in implications is employing them as a selection mechanism for variable values. For example, for variable V1 the preference scale cited earlier in (10) is relevant, th we may decide to include in the analysis only the values with scale va 'obligatory' or 'preferred' for any language.

4.4.8.3 Syntagmatic conditions

Like scale values, syntagmatic conditions may be added to factors, with t same type of effect as described in section 2. Given their often idiosyncrat

character, however, relevancy values will generally be rather low.
Further, the mere presence or absence of conditions may be used as a selection mechanism, without any further specification of the nature of the condition. This leads to the following type of representations:

```
(48)  a.  V1=val_1                   ->  V2=val_2a
      b.  V1=val_1    <CONDITION>    ->  V2=val_2b
```

A more sophisticated treatment, that deals with the contents of conditions, has not yet been implemented as part of the analysis process. It will become more interesting when an integrated data base will be available containing variables stemming from several subdomains, that may appear in the syntagmatic conditions of variables from other domains. Apart from (implicit) implications between variables stemming from different linguistic domains, such cros-domain conditions explicitly build links between parts of the grammar.

4.4.8.4 Missing values

Missing values for variables may be entered via the QTP program or be defined by the user. In the QTP program there are two types of missing values: ? (= value not provided) and a blank (= variable not relevant). The user may specify a set of real values for the variable set that are defined as 'missing' in the domain definition file and the data base (see (10c) above).
The system defined missing value ? is left out of any analysis (i.e. there will be no implications for such values). The number of missing values is reported in the output for any selected variable. In case a variable is irrelevant for some language, the latter is left out of corresponding implications by definition. As far as the user defined missing values are concerned, these may optionally be treated either as 'normal' values or as ?. In case of a scale being applied to some variable, a missing scale value for some language will be treated by the program as a separate option for that scale. This means that it may be manipulated by the user in the same way as 'real' scale values may.

4.4.8.5 Verification

Apart from inferenceing rules from the data base, the LINFER program may be used to verify simple and complex implications provided by the user. In this way, any implication over the variables in the data base one may want to investigate may be tested. Also, universals found in typological literature, that may be formulated in terms of implications over variables and values of the data base may be verified. Especially the latter option may be of interest. As an example, I took the following related universals from (Dryer 1992, 56), that are based on his 625 language sample:

(49) If a language is prepositional, it will employ clause-initial adverbial subordinators

(49) could be translated to variables V0_2 (adposition) and V19_2 (location subordinator) of the word order data base, and came out in the following fashion:

	V2=val2	V2#val2	total
V1=val1	18 (=appl)	2 (=cntrex)	20 (=rel)
V1#val1	1	9	10
total	19 (=cov)	11	30 (=tot)

8. At face value, it may sound counterintuitive that an implication that applies to, say, 3 languages should be called a universal. However, if it is found in a relatively small language sample, e.g. 30 or 50, that is supposed to be representative for the phenomena under analysis, any subset may, in fact, represent several hundreds of languages. In the case of acknowledged isolates we may even deal with a phenomenon that is restricted to one single language, while it may still broaden our insight into the concept of 'possible human language'.

9. An alternative notation for (28) is:
(28') V1=v_1 AND V2=v_2 AND V3=v_3 -> V4=v_4
which makes more apparent the role of the AND operator.

10. Formally, this is not a restriction since NOT operators may always be lowered in any expression in propositional logic.

11. We adopt the < rather than the > to represent such hierarchies since the latter may, wrongly, be interpreted as an implication sign. In fact, (38) corresponds to the following chain:
(38') valn -> ... -> val3 -> val2 -> val1

12. In the version of SPSS currently in use - SPSS/PC version 3.01 - the maximum number of variables that may be included in a cluster analysis is 600. The cluster method built into LINFER includes any number of implications, but does not give the sort of hierarchical ordered results shown below.

13. In his note 8 Dryer in fact refines universal (49) to the extent that languages for which it is not relevant, i.e. that have no subordinators, are explicitly excluded. Also, it is turned into a tendency by including the languages that employ both types of adposition, but prefer prepositions, and languages that employ both clause-initial and clause-final subordinators, but have a preference for the first.

REFERENCES

Berlin, B. & P. Kay (1969).
Basic Color Terms: their Universality and Evolution. Berkeley: University of California Press.

Comrie, B. (1981).
Language Universals and Linguistic Typology. Chicago: University of Chicago Press.

Croft, B. (1990).
Typology and Universals. Cambridge: Cambridge University Press.

Dik, S.C. (1989).
The Theory of Functional Grammar. Dordrecht: Foris.

Dryer, M.S. (1987).
A Statistical Study of Word Order Universals. Final report for Social Sciences and Humanities Research Council of Canada Research Grant. University of Alberta.

Dryer, M.S. (1992).
Adverbial Subordinators and Word Order Asymmetries. In: ESF Working Paper II/2, 50-67.

Greenberg. J. (1963).
Some Universals of Grammar with Particular Reference to the Order of Meaningful Elements. In J.H. Greenberg (ed.) Universals of Language. Cambridge (Mass.): MIT Press, 58-90.

Hawkins, J. (1983).
Word Order Universals. New York: Academic Press.

Rijkhoff, J., D. Bakker, K. Hengeveld & P. Kahrel (1992).
A Method of Language Sampling. Studies in Language (to appear).

Ruhlen, M. (1987).
A Guide to the World's Languages.Vol. 1: Classification. London: Edward Arnold.

Siewierska, A. (1988).
Word Order Rules. London: Croom Helm.

SPSS/PC. Base Manual and Advanced Statistics.

Van der Steen, G.J. (1987).
A Program Generator for Recognition, Parsing and Transduction with Syntactic Patterns. PhD thesis, University of Utrecht.

APPENDIX A: FORMAL DEFINITION OF THE DSL LANGUAGE

The formalism is based on context-free rewriting rules with extensions that give it transformational power, as used by the parser-compiler system Parspat (van der Steen 198x). For DSL, only context-free rules are used.

```
! DSL: Domain Structuring Language !
!----------------------------------!

DSL :: Domain  |  Data_Base.


! 1. QUESTIONNAIRE FILE !
!-----------------------!

Domain :: Header, [Scales], [Miss_Val], [Cond_Vars], Var_Descs, Footer.

Header :: Text, CR.

Scales :: Scale, [Scales].
Scale  :: 'scale', Name, Eq, Value_List, CR.

Cond_Vars :: 'condition_variables', Eq, Variable_List, CR.

Var_Descs :: Var_Desc, [Var_Descs].
Var_Desc  :: Astrx, Name, [Lbr, Ext_String, Rbr], Ext_String,
                   CR, Parameters, CR, Values.

Footer :: Astrx, CR.

Parameters :: 'parameters', Eq, Parameter_List.
Parameter_List :: Lbr, Pars, Rbr.
Parameter_List :: Lbr, Rbr.
Pars :: Par, [Pars].
Par  :: 'scale', Eq, Name.
Par  :: 'paradigmatic_condition', Eq, Implication.
Par  :: 'syntagmatic_condition'.
Par  :: 'open'.
Par  :: 'multi', Eq, Number.
Par  :: 'hierarchical'.

Values :: Value, CR, [Values].
Value  :: Lower, Stop, Lable.

! 2. DATA BASE !
!--------------!

Data_Base :: [Miss_Val], Domain_Variables, Language_Descriptors.

Domain_Variables :: Domain_Variable, [Domain_Variables].
Domain_Variable  :: Name, Eq, Ext_String, CR.

Language_Descriptors :: Language_Descriptor,
                               [Language_Descriptors].
```

```
Language_Descriptor  :: Dlr, Number, Name, Number, CR,
                              Var_Val_List.
Language_Descriptor  :: Dlr, Number, Name, Astrx, CR.

Var_Val_List :: Var_Val_Set, [Var_Val_List].

Var_Val_Set  :: Name, Eq, Val_Sets, CR.
Var_Val_Set  :: Name, Eq, Qst, CR.

Val_Sets :: Val_Set, [Bcksl, Val_Sets].
Val_Set  :: Lable, [Hyph, Scale_Cond].

Scale_Cond :: Lsbr, Scale_Value, Comma, Syntagmatic_Condition,
                 Rsbr.

Scale_Value :: Lbr, [Lable], Rbr.

Syntagmatic_Condition :: Lbr, [Condition], Rbr.



! 3. SHARED RULES !
!----------------!

Miss_Val :: 'missing_values', Eq, Value_List, CR.

Variable_List :: Lbr, Names, Rbr.

Value_List :: Lbr, Lables, Rbr.

Implication :: Lsbr, Condition, Arrow, Name, Rsbr.

Condition :: Cond, [L_Op, Condition].
Condition :: Lbr, Condition, Rbr.
Cond :: Name, E_Op, Lable.

Names :: Name, [Comma, Names].
Name  :: Upper, [String].

Number :: Digit, [Number].

Lables :: Lable, [Comma, Lables].
Lable  :: Letter, [String].
Lable  :: Digit, [String].

String :: String_Symbol, [String].
String_Symbol :: Letter | Digit | Und | Slash.

Ext_String :: Ext_Symbol, [Ext_String].
Ext_Symbol :: Letter | Digit | Und | Amp | Eq | Ueq | Hyph |
              Stop | Comma | Slash | Qst | Excl | Space.

Text :: Quote, Text_String, Quote.
Text_String :: Text_Symbol, [Text_String].
Text_Symbol :: Ext_Symbol | CR.
```

```
! 4. TERMINALS !
!--------------!

L_Op :: 'AND' | 'OR'.
E_Op :: Eq | Ueq.

Letter :: Upper | Lower.
Upper :: A..Z.
Lower :: a..z.
Digit :: 0..9.

Quote :: '"'.
Comma :: ','.
Stop  :: '.'.
Eq    :: '='.
Ueq   :: '#'.
Lbr   :: '('.
Rbr   :: ')'.
Lsbr  :: '['.
Rsbr  :: ']'.
Und   :: '_'.
Hyph  :: '-'.
Amp   :: '&'.
Qst   :: '?'.
Excl  :: '!'.
Astrx :: '*'.
Space :: ' '.
Arrow :: '>'.
Slash :: '/'.
Bcksl :: '\'.
Dlr   :: '$'.
```

APPENDIX B: QST FILE FOR THE WORD ORDER DOMAIN

N.B. only a few, more or less representative variables are displayed here. Actually, the domain is coded in terms of around 225 variables.

```
'EUROTYP
'WORD ORDER QUESTIONNAIRE
'Version 1.0
'Date Nov-20-91


scale Preference=[oblig, pref, non_pref, rare]
scale Restrict=[fav, restr]
scale Style=[colloqial, literary, dialectic, formal, spoken,
              written]
scale Obligat=[obl, opt]

missing_values=[not_known, not_clear, not_sure]


*V0_1 (= 0.1) greenbergian classif
- open *
a. SOV
b. SOV/rigid
c. SOV/free
d. SVO
e. SVO/free
f. SVO/V2
g. VSO
h. VOS
i. OVS
j. OSV
k. V1
l. V2
m. free
*V1_2 (= 1.2) agent topic
- multi=5 open scale=preference *
a. SOV
b. SVO
c. VSO
d. VOS
e. OVS
f. OSV
g. VprtVSO
h. LdS
i. VAg
j. XVSO
*V2_1 (= 2.1 - 2.4 ) order Si
- scale=preference cond_synt multi=2 open *
a. SiV
b. VSi
```

```
*V29_8a (= ) nominal cases
-  cond_synt cond_out=[(V29_8a=none > V29_9)] scale=restrict multi=10 *
a. none
b. obl
c. nom
d. acc
e. dat
f. gen
g. instr
h. loc
i. voc
j. part
k. erg
l. abs
*V29_8b (= ) local cases
- cond_synt open multi=15 *
a. none
b. iness
c. elat
d. illat
e. adess
f. abl
g. all
h. term
i. subess
j. adlat
k. superess
l. delat
m. sublat
n. postess
o. adelat
p. subelat
q. postelat
r. superelat
s. superdir
t. inelat
u. lative
*V29_8c (= ) other adverbial cases
- open cond_synt multi=15 *
a. none
b. ess
c. comit
d. carit
e. trans
f. abess
g. instruct
i. sociat
j. destin
k. motiv
l. prolat
m. distr
n. process
o. modal
p. egress
q. precl
r. consec
```

APPENDIX C: PART OF THE WORD ORDER DATA BASE

N.B. only a small section is shown. Currently, the data base contains data on 44 languages coded in about 225 variables, of which an average of 175 are relevant per language.

```
$ Dutch
V0_1=SVO/V2
V0_2=prefprep
V1_1=SVO
V1_2=SVO - [(Preference=pref) , ( )]\
       VSO - [(Preference=non_pref) , ( )]\
       VOS - [(Preference=non_pref) , ( )]\
       OVS - [(Preference=non_pref) , ( )]
V1_4=SpassPeriphAg - [(Preference=oblig) , ( )]
V1_6=noeffect
V1_7=noeffect
V1_8=nocl
V2_1=VSi - [(Obligat=obl) , ((Pattern=Expletp1) OR
                (Pattern=Advp1)) , (Preference=pref) ,
                    ((DefS=indef) OR (Wght=heavy))]\
       SiV - [(Preference=pref) , ((DefS=def) OR (WghtS=light))]
V2_2=LcN - [(Preference=pref) , ( )]\
       ExplcNL - [(Preference=pref) , ( )]\
       LcExplN
V3_0=prep\zero
V3_1=RnomPnom\PproRpro\PproRadp
V3_2=PRadp
V3_3=none
V3_4=RecBen
V3_5=LadpIadp
V4_1=OVS\SVO
V4_2=AdvAuxSOV\AdvVSO
V4_3=OVS\SVO
V4_5a=no
V4_5=subclause\AdvAuxSOV
V4_5p=?
V4_6=OAuxSV
V4_6p=?
V4_7=quest\XVSO
V4_7p=?
V4_8=impossb
V4_9=prag\Owh
V4_9p=Ofoc
V4_10=?
V4_10p=?
V4_11=?
V4_12=SstrVO
V4_12a=P1
V4_13=OstrVS
V4_13a=P1
V4_14=SVOPPstr\PPstrVSO
V4_14a=P1\final
V4_15=no
V14_1=no
V14_4=asinSubCl
```

APPENDIX D: SPSS VERSION OF PART OF THE WORD ORDER DATA BASE

N.B. This file is generated automatically by the TRANS module.

```
DATA LIST FREE/LNGNR
 V0_1 V0_2 V1_1 V1_2_A V1_2_B V1_2_C V1_2_D V1_2_E
 V1_4_A V1_4_B V1_4_C V1_4_D V1_4_E V1_6 V1_7 V1_8_A V1_8_B
 V2_1_A V2_1_B V2_2_A V2_2_B V2_2_C V2_2_D

VARIABLE LABELS LNGNR "language number"
 /V0_1 "greenbergian classif"
 /V0_2 "adposition"
 /V1_1 "allnew order"
 /V1_2_A "agent topic A"
 /V1_2_B "agent topic B"
 /V1_2_C "agent topic C"
 /V1_2_D "agent topic D"
 /V1_2_E "agent topic E"
 /V1_4_A "patient topic A"
 /V1_4_B "patient topic B"
 /V1_4_C "patient topic C"
 /V1_4_D "patient topic D"
 /V1_4_E "patient topic E"
 /V1_6 "full pro O"
 /V1_7 "unstress pro O"
 /V1_8_A "clitic O A"
 /V1_8_B "clitic O B"
 /V2_1_A "order Si A"
 /V2_1_B "order Si B"
 /V2_2_A "order existential A"
 /V2_2_B "order existential B"
 /V2_2_C "order existential C"
 /V2_2_D "order existential D"

VALUE LABELS LNGNR
    27 "Abxaz"
   119 "Albanian"
   483 "Basque"
   684 "Breton"
   710 "Bulgarian"
   767 "Byelorussian"
   817 "Catalan"
   950 "Chuvash"
   955 "Classical-Greek"
  1015 "Czech"
  1047 "Danish"
  1052 "Dargwa"
  1093 "Dido"
  1178 "Dutch"
  1249 "English"
  1265 "Estonian"
  1299 "Finnish"
  1321 "French"
  1323 "Frisian"
  1434 "Georgian"
 1437 "German"
 1493 "Greek"
 1681 "Hungarian"
 1772 "Italian"
 2211 "Komi"
 2497 "Lezgi"
 2527 "Lithuanian"
 2632 "Macedonian"
 2701 "Maltese"
 2802 "Mari"
 3069 "Mordvin"
 3436 "Northern-Saami"
 3444 "Norwegian"
 3801 "Polish"
 3963 "Russian"
 4128 "Serbo-Croatian"
 4254 "Slovak"
 4255 "Slovene"
 4331 "Spanish"
 4382 "Swedish"
 4767 "Udmurt"
 4778 "Ukrainian"
 4813 "Upper-Sorbian"
 4997 "Welsh"
 /V0_0a 1 "Indo_European"
 /V0_0b 1 "Slavic"
 /V0_1 1 "SOV" 2 "SOV/free" 3 "SVO" 4 "SVO/free" 5 "SVO/V2"
 6 "VSO" 7 "V1" 8 "V2" 9 "free"
 /V0_2 1 "prep" 2 "post" 3 "prefprep" 4 "prefpost" 5 "bothequal"
 /V1_1 1 "SVO" 2 "SOV" 3 "VSO" 4 "VprtVSO"
 /V1_2_A TO V1_2_E 1 "SOV" 2 "SVO" 3 "VSO" 4 "VOS" 5 "OVS"
 6 "OSV" 7 "VprtVSO" 8 "LdS" 9 "VAg" 10 "XSVO"
 /V1_4_A TO V1_4_E 1 "OVS" 2 "OSV" 3 "SpassPeriphAg" 4 "SpassSynthAg"
 5 "VpassSAg" 6 "Ld" 7 "SVO" 8 "SOV" 9 "VOS" 10 "AgPassPeriphS"
 11 "OSVbeAux" 12 "VPat" 13 "SVbeAuxO"
 /V1_6 1 "noeffect" 2 "noVO" 3 "prefVO" 4 "notwithVanalyt"
 5 "ClDoub" 6 "2sgplObefore1sgS" 7 "prefVinit_Vfin"
 /V1_7 1 "noeffect" 2 "VOnegnotVnegVO" 3 "prefOV" 4 "notP1"
 5 "irr"
 /V1_8_A TO V1_8_B 1 "nocl" 2 "proclV" 3 "encl" 4 "VSproclVoptencl"
 5 "not_known" 6 "2nd"
 /V2_1_A TO V2_1_B 1 "SiV" 2 "VSi"
 /V2_2_A TO V2_2_D 1 "NcL" 2 "NLc" 3 "LNc" 4 "cNL" 5 "LcN"
 6 "ExplcNL" 7 "LcExplN" 8 "ExplNL" 9 "LN"

MISSING VALUES V0_0a TO V2_2_D (0).
BEGIN DATA.

 1178
  0  0  5  3  1  2  3  4  5  0  3  0  0  0  0  1  1  1  0  2
  1  5  6


END DATA.
```

APPENDIX E: SOME DETAILS ON THE PROGRAMS

1. QTP

Goal: data entry of questionnaire data
Programming language: Pascal
Source size: 3000 lines
Versions: MS/DOS; VAX
Minimum memory necessary: 640K

2. TRANS

Goal: transformation of data files
Programming language: Pascal
Source size: 2000 lines
Versions: VAX
Minimum memory necessary: 1M

3. DBL

Goal: run language data base
Programming language: Pascal
Source size: 5500 lines
Versions: VAX
Minimum memory necessary: 4M

4. LINFER

Goal: inference of linguistic rules
Programming language: Pascal
Source size: 6000 lines
Versions: MS/DOS; VAX
Minimum memory necessary: MS/DOS 640K; VAX 2M

Acknowledgement:

For sorting, the programs use Hoare's QuickSort method, adapted from the original Fortran version as programmed by Karel Sprenger, Frank Koperdraat and Hans Dekker.