6. Coding, storage, retrieval and analysis of analytical linguistic data

In what follows the term analytical data will be used to denote all systematic and linguistically relevant observations on linguistic entities, such as words, clauses or stretches of discourse, for a specific language, on any level of description, either in terms of a specific theory or in terms of a more general descriptive framework. Some examples of such data are:

a. the basic order in main and subordinate clauses
b. the way different tenses are expressed
c. the position of the complimentizer in relation to the clausal category it selects
d. the occurrence and the type of vowel harmony
e. the occurrence of pronominal politeness forms and whether it is marked or unmarked in spoken language.

Much work in language typology is based on collections of this type of data. Often they are gathered by means of a questionnaire such as the ones described in section 5 of the guidelines. Even before a questionnaire is constructed, the researcher should establish a (tentative) model of the linguistic domain concerned. From such a model, the actual questions should be derived, and structured. The questionnaire results may then be interpreted in terms of the initial model. This may eventually lead to modifications and refinements in the model. If no such modelling had been done before the questionnaire was constructed, possibly since the knowledge of the domain was still too sketchy, the modelling may be done precisely on the basis of the incoming questionnaire results.

If the properties of the model, composed prior to the questionnaire or based on its results, comply to certain minimal formal standards, then the incoming data may be formalized as well, and processed by a computer. For this, standard data base and data analysis programs may be used. Well-known examples of the former are D-Base, Oracle and Shoebox. Section 10 of the guidelines gives a standard format. Data analysis may be performed with statistical packages such as SPSS. The drawback of these standard programs is that they provide only very general facilities, and are not tuned to the more specific needs of the linguist. The data formats that are required are often quite 'unnatural', i.e. not of the form in which observations often are formulated by linguists. A convenient means for doing so is presented below.

First, section 6.1 discusses the features of a theory-independent system for representing analytical data. And in section 6.2 these features are implemented in a formal descriptive language called Domain Structuring Language (DSL), that we think provides a more natural way for linguistic data representation than the formalisms generally used in data processing. A set of computer programs, based on DSL, has been devised that may be used for coding, storing and analyzing such data, taking over part of the work that has to be done by the linguist. They implement a number of data analysis techniques that may not be found among the standard software packages. For standard tasks, interfaces have been built in that give access to some of the generally available packages. The programs are briefly described in section 10.1. A full

description of the possibilities is given in working papers 2-3 and 2-5 of the Theme Group on Constituent Order.

6.1. Coding a linguistic domain

In empirical sciences, a descriptive domain is often structured by way of variables and values. Variables provide a syntagmatic structuring to the domain, i.e. they give the dimensions that exist in parallel. Any variable has a set of values assigned to it, that give a paradigmatic structuring to the domain. In the simplest form, all variables are thought to be relevant for all cases, i.e. all entities that are the object of analysis, such as languages for linguistics. Variable values are mutually exclusive, i.e. any case selects precisely one value for each variable. A schematic version of this representation scheme is the data matrix of figure 1, with the cases on the vertical and the variables on the horizontal dimension. Every cell contains precisely one value, i.e. the score for the given case in the row in regard to the variable in the column.

	variable 1	variable 2	variable 3
case 1	value 1_1	value 1_2	value 1_3
case 2	value 2_1	value 2_2	value 2_3
case 3	value 3_1	value 3_2	value 3_3

Figure 1. Data matrix

In order to ensure that such a matrix will be complete, special values may be employed to code the fact that, for a specific case, no value is known for some variable, or that some variable is not relevant altogether. These are commonly called 'missing values', and may receive special treatment under analysis. Although, formally, this data matrix representation method is rich enough to code any type of analytical data, it does so in a manner that diverges from the way linguistic observations are normally formulated.

To enable the representation of language facts in a way closely corresponding to the actual practice of linguists, the above simplified variable - value format has been enriched by several conventions. Each of these will be discussed briefly in turn.

a. Multiple values

The canonical way of coding language phenomena the realization of which either typically does or may involve a set of options is to set up a separate variable for each option and supply it with a yes/no value.

Needless to say, this can be very cumbersome. When dealing with a phenomenon with numerous options, for example, the type of case distinctions made, such a solution would require over 40 variables.

What is therefore necessary is to allow variables to be associated with multiple values

ranging from two to whatever number is required.

b. Structured values

It is generally impossible to predict all the values for all the variables in terms of which one has structured a given domain. It is therefore advantageous to use values which can be potentially concatenated to form new values. Moreover, if some of the values used have internal structure, i.e. they are decomposable, it may be possible to form new values from the composite parts of the old. This may considerably facilitate the drawing of generalizations over values. For example, given the three values SOV, OSV and OVS each of which can be decomposed into S, O and V, we can generalize over OV (all 3 of the values), SV (2) or OS order (2).

c. Scales

Often, linguistic data, and more precisely values for variables, may be subject to scaling along some dimmension. For example, in a number of languages the location of adverbials of setting is typically sentence initial or sentence final. Clause internal placement would be considered as marked. This phenomenon could be coded in several ways. We could have separate variables for the unmarked and the marked location. We could have one variable and then concatenate 'marked' and 'unmarked' to the location values. We extended the variable-value scheme with the possibility of associating the values of particular variables with additional scalar values. The above example on adverbials may then be coded by providing a markedness scale for the variable 'location of adverbial of setting' and allowing one to specify which of the values chosen for this variable such as 'initial', 'internal' and 'final' are marked or unmarked.

d. Conditions

In a variable-value scheme, we may have to deal with two types of conditions: syntagmatic and paradigmatic ones. Syntagmatic conditions occur when a language has several values for one variable, each occurring under different linguistic circumstances. For example, the relationship between the possessor and possessed in English may be coded either by a preposition between the two or by a genitive enclitic on the possessor. There are no specific (categorical) conditions on the use of the first construction. The use of the latter, however, is restricted to animate nouns that have to be relatively short. Instead of coding such conditions in the values, they can be specified as a condition on a value.

Paradigmatic conditions involve dependencies between values. It is sometimes the case that the relevance of some variable for a language is dependent on the value for some other variable. Cf. when a language has no articles, variables coding their precise aspects, such as their formand location, are irrelevant for that language. To capture this fact it is desirable to have a variable referring to the existence of articles and to place this variable before the ones pertaining to specific features of articles or their use. Thus if the first variable receives the value "no" all the dependent variables will be defined as irrelevant.

These extensions lead to a four-dimensional way of structuring a linguistic domain, in terms of variables, values, scales and conditions.A schematic representation may look as follows:

Figure 2. Graphic representation of a linguistic domain (to be filled in)

6.2. The representational language

The DSL language provides the means of structuring and coding a linguistic domain precisely in terms of the four dimensions mentioned above. A complete formal definition is found in appendix A. Here, only some representative examples of its use for the coding of information will be given. The information stems from the domain of word order, case systems and adverbials. The overall structure of a domain definition is as follows:

(1) a. Domain description
b. Scales
c. Missing values
d. Condition variables
e. Domain variable descriptors

(1a) is a short text describing the domain, meant for identification purposes only. (1b) are the definitions of the relevant scales. (1c) provides a list of those values that should be considered as 'missing' for any of the variables in the domain. The condition variables (1d) do not necessarily belong to the domain proper, and may be used in syntagmatic conditions. For the word order data questionnaire, (1a) through (1d) look as follows (abbreviated):

(2) a. Word Order Data, Version 1.1 Nov92
b. Scales: Preference=(oblig, pref, non_pref, imposs)
Restrictive=(fav, restr)
c. Missing: (not_known, not_clear, not_present)
d. External: Position1=(expletive, adverb)
DefSubj=(def, indef)

Variable Position1 in the list of externals may be used in syntagmatic conditions to represent what is in the first position; variable DefSubj will represent the definiteness status of the subject.

The basic entities of DSL are the domain variable descriptors (1e), that give a formal description of the respective variables that paradigmatically code the domain concerned. They are provided as an ordered list of individual descriptors. A variable descriptor is built up in the following way:

(3) 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 (cf. Var207). The reference is an (optional) indication of the source of the value of the variable, such as a particular set of questions in a questionnaire. It is basically a convenience for checking the original source of the data

in a questionnaire. The variable label is a description of the meaning of the variable, for instance, 'order of the possessor and possessed', 'existence of impersonal passive', etc. Parameters characterize the variable in terms of its type (e.g. multi-valued; complex values), associated scales, and conditions. The value set provides an exhaustive enumeration of the values for the variable. Some of the above features are illustrated in (4):

(4) 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. Any one of them may be associated with a value from the scale 'preference'(see (2b)). The presence of the 'syntagmatic_condition' parameter allows a syntagmatic condition to be stated for any value chosen for the variable for some language. Definition (4) gives the possible values of domain variable V2.1 along all relevant dimensions. For a particular language that is in the sample, the values for a variable are coded according to the following syntax:

(5) var = value_1 - [(scale=value) , (syntagm_condition)] /
value_2 - [(scale=value) , (syntagm_condition)] /
...

When scale and condition are irrelevant the part after the hyphen may be left out. Syntagmatic conditions are boolean expressions over variables and values from the set of condition variables. In conditions, the operators AND and OR may be used, as well as unlimited embedding by means of brackets. In the simple expressions, both the equality operator = and the unequality operator # may be used. We may end up with the following value description for variable V2.1 for some specific language L. In other words: these are the contents of the cell on row L and column V2.1 of the data matrix:

(6) V2.1= VSi -[(preference=obligatory) ,
((Pos1=expletive) OR (Pos1=adverb) OR (DefS=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. An example of a paradigmatic condition is given in (7). Note that, while syntagmatic conditions are part of the data for a specific language, paradigmatic conditions are on variable descriptors:

(7) V29.8a (Q29.3 - Q29.4)
Nominal cases
parameters= (multi=10, paradigm_condition=[V29.8a=none > V29.9])

Thus, if V29.8a has value 'none' for some language, the variable immediately dependent on it is V29.9, not the default one, i.e. the variable of which the descriptor immediately follows that of V29.8a, say V29.8b. This compares to the relationship between variables V1, V2 and V3 in figure 2 above. In this type of paradigmatic condition, before the 'greater than' sign we have a simple equality or unequality expression for the current variable. After the sign we find a variable label. There may be more than one such paradigmatic condition in a variable descriptor.

A second type of paradigmatic condition on variable V consists of an implication with a boolean expression over variables higher than V for its premise and a subset of the values of V for its conclusion. Assuming that variables V7 and V8 are higher than V9, i.e. their descriptors come before that of V9, we could have the following for paradigmatic condition V9:

(8) paradigm_condition=[(V7 = val7a AND V8 # val8c) -> (val9b, val9d)]

meaning that if V7 has value val7a and V8 does not have value val8c then the selectable values for V9 should be restricted to the subset val9b and val9d. The latter version of the paradigmatic condition could be called an input condition, the former one an output condition: For any language, the former determines the options for a variable before it is assessed, in the context of the values determined for higher variables. The latter determines the configuration of variables lower than some variable after its value has been determined. A last option in the parameter field of a variable descriptor is the 'open' option. This signals that, apart from the predetermined set of values, the variable may be assigned any other value that has not been specified in advance.

Back to index