6 Feature descriptions

In the previous chapter, we talked about sets of feature-value pairs, which can be used to describe linguistic objects. In this chapter, we will introduce feature descriptions which play a role in theories such as LFG, HPSG, Construction Grammar, versions of Categorial Grammar and TAG (and even some formalizations of Minimalist theories (Veenstra 1998)). This chapter will therefore lay some of the groundwork for the chapters to follow.

Feature structures are complex entities which can model properties of a linguistic object. Linguists mostly work with feature descriptions which describe only parts of a given feature structure. The difference between models and descriptions will be explained in more detail in Section 6.7.

Alternative terms for feature structures are:

- feature-value structure
- attribute-value structure

Other terms for feature description are the following:

- *attribute-value matrix* (AVM)
- *feature matrix*

In what follows, I will restrict the discussion to the absolutely necessary details in order to keep the formal part of the book as short as possible. I refer the interested reader to Shieber (1986), Pollard & Sag (1987: Chapter 2), Johnson (1988), Carpenter (1992), King (1994) and Richter (2004). Shieber's book is an accessible introduction to Unification Grammars. The works by King and Richter, which introduce important foundations for HPSG, would most probably not be accessible for those without a good grounding in mathematics. However, it is important to know that these works exist and that the corresponding linguistic theory is build on a solid foundation.

6.1 Feature descriptions

When describing linguistic signs, we have to say something about their properties. For a noun, we can say that it has case, gender, number and person features. For a word such as *Mannes* 'man', we can say that these features have the values *genitive*, *masculine*, *singular* and *3*. If we were to write these as a list of feature-value pairs, we would arrive at the following feature description:

(1) Feature-value pair for *Mannes*:

CASE	*genitive*
GENDER	*masculine*
NUMBER	*singular*
PERSON	*3*

It is possible to describe a variety of different things using feature descriptions. For example, we can describe a person as in (2):

(2)

FIRSTNAME	*max*
LASTNAME	*meier*
DATE-OF-BIRTH	*10.10.1985*

People are related to other people – a fact that can also be expressed in feature-value pairs. For example, the fact that Max Meier has a father called Peter Meier can be captured by expanding (2) as follows:

(3)

FIRSTNAME	*max*	
LASTNAME	*meier*	
DATE-OF-BIRTH	*10.10.1985*	
FATHER	FIRSTNAME	*peter*
	LASTNAME	*meier*
	DATE-OF-BIRTH	*10.05.1960*
	FATHER	…
	MOTHER	…
MOTHER	…	

The value of the FATHER feature is another feature description containing the same features as (2).

In feature descriptions, a *path* is a sequence of features which immediately follow each other. The *value of a path* is the feature description at the end of the path. Therefore, the value of FATHER|DATE-OF-BIRTH is *10.05.1960*.

One can think of many different features that could be included in representations such as (3). One may wonder how to integrate information about offspring into (3).

An obvious solution would be to add features for DAUGHTER und SON:

(4)

FIRSTNAME	*max*
LASTNAME	*meier*
DATE-OF-BIRTH	*10.10.1985*
FATHER	…
MOTHER	…
DAUGHTER	…

This solution is not satisfactory as it is not immediately clear how one could describe a person with several daughters. Should one really introduce features such as DAUGHTER-1 or DAUGHTER-3?

(5)
$$\begin{bmatrix}
\textsc{firstname} & \textit{max} \\
\textsc{lastname} & \textit{meier} \\
\textsc{date-of-birth} & \textit{10.10.1985} \\
\textsc{father} & \ldots \\
\textsc{mother} & \ldots \\
\textsc{daughter-1} & \ldots \\
\textsc{daughter-2} & \ldots \\
\textsc{daughter-3} & \ldots
\end{bmatrix}$$

How many features do we want to assume? Where is the limit? What would the value of DAUGHTER-32 be?

For this case, it makes much more sense to use a list. Lists are indicated with angle brackets. Any number of elements can occur between these brackets. A special case is when no element occurs between the brackets. A list with no elements is also called *empty list*. In the following example, Max Meier has a daughter called Clara, who herself has no daughter.

(6)
$$\begin{bmatrix}
\textsc{firstname} & \textit{max} \\
\textsc{lastname} & \textit{meier} \\
\textsc{date-of-birth} & \textit{10.10.1985} \\
\textsc{father} & \ldots \\
\textsc{mother} & \ldots \\
\textsc{daughter} & \left\langle \begin{bmatrix}
\textsc{firstname} & \textit{clara} \\
\textsc{lastname} & \textit{meier} \\
\textsc{date-of-birth} & \textit{10.10.2004} \\
\textsc{father} & \ldots \\
\textsc{mother} & \ldots \\
\textsc{daughter} & \langle\rangle
\end{bmatrix} \right\rangle
\end{bmatrix}$$

Now, we are left with the question of sons. Should we add another list for sons? Do we want to differentiate between sons and daughters? It is certainly the case that the gender of the children is an important property, but these are properties of the objects themselves, since every person has a gender. The description in (7) therefore offers a more adequate representation.

At this point, one could ask why the parents are not included in a list as well. In fact, we find similar questions also in linguistic works: how is information best organized for the job at hand? One could argue for the representation of descriptions of the parents under separate features, by pointing out that with such a representation it is possible to make certain claims about a mother or father without having to necessarily search for the respective descriptions in a list.

(7)
$$\begin{bmatrix}
\textsc{firstname} & \textit{max} \\
\textsc{lastname} & \textit{meier} \\
\textsc{date-of-birth} & \textit{10.10.1985} \\
\textsc{gender} & \textbf{\textit{male}} \\
\textsc{father} & \ldots \\
\textsc{mother} & \ldots \\
\textsc{children} & \left\langle \begin{bmatrix}
\textsc{firstname} & \textit{clara} \\
\textsc{lastname} & \textit{meier} \\
\textsc{date-of-birth} & \textit{10.10.2004} \\
\textsc{gender} & \textbf{\textit{female}} \\
\textsc{father} & \ldots \\
\textsc{mother} & \ldots \\
\textsc{children} & \langle\rangle
\end{bmatrix} \right\rangle
\end{bmatrix}$$

If the order of the elements is irrelevant, then we could use sets rather than lists. Sets are written inside curly brackets.[1]

6.2 Types

In the previous section, we introduced feature descriptions consisting of feature-value pairs and showed that it makes sense to allow for complex values for features. In this section, feature descriptions will be augmented to include types. Feature descriptions which are assigned a type are also called *typed feature descriptions*. Types say something about which features can or must belong to a particular structure. The description previously discussed describes an object of the type *person*.

(8)
$$\begin{bmatrix}
\textsc{firstname} & \textit{max} \\
\textsc{lastname} & \textit{meier} \\
\textsc{date-of-birth} & \textit{10.10.1985} \\
\textsc{gender} & \textit{male} \\
\textsc{father} & \ldots \\
\textsc{mother} & \ldots \\
\textsc{children} & \langle \ldots, \ldots \rangle \\
\textit{person} &
\end{bmatrix}$$

Types are written in *italics*.

The specification of a type determines which properties a modelled object has. It is then only possible for a theorie to say something about these properties. Properties such

[1] The definition of a set requires many technicalities. In this book, sets would be used for the collection of semantic information only. This can be done equally well using lists, which is why I do not introduce sets here and instead use lists.

as OPERATING VOLTAGE are not relevant for objects of the type *person*. If we know the type of a given object, then we also know that this object must have certain properties even if we do not yet know their exact values. In this way, (9) is still a description of Max Meier even though it does not contain any information about Max' date of birth:

(9)
$$\begin{bmatrix} \text{FIRSTNAME} & \textit{max} \\ \text{LASTNAME} & \textit{meier} \\ \text{GENDER} & \textit{male} \\ \textit{person} & \end{bmatrix}$$

We know, however, that Max Meier must have been born on some day since this is a description of the type *person*. The question *What is Max' date of birth?* makes sense for a structure such as (9) in a way that the question *Which operating voltage does Max have?* does not. If we know that an object is of the type *person*, then we have the following basic structure:

(10)
$$\begin{bmatrix} \text{FIRSTNAME} & \textit{firstname} \\ \text{LASTNAME} & \textit{lastname} \\ \text{DATE-OF-BIRTH} & \textit{date} \\ \text{GENDER} & \textit{gender} \\ \text{FATHER} & \textit{person} \\ \text{MOTHER} & \textit{person} \\ \text{CHILDREN} & \textit{list of person} \\ \textit{person} & \end{bmatrix}$$

In (10) and (9), the values of features such as FIRSTNAME are in italics. These values are also types. They are different from types such as *person*, however, as no features belong to them. These kinds of types are called *atomic*.

Types are organized into hierarchies. It is possible to define the subtypes *woman* and *man* for *person*. These would determine the gender of a given object. (11) shows the feature structure for the type *woman*, which is analogous to that of *man*.

(11)
$$\begin{bmatrix} \text{FIRSTNAME} & \textit{firstname} \\ \text{LASTNAME} & \textit{lastname} \\ \text{DATE-OF-BIRTH} & \textit{date} \\ \text{GENDER} & \textit{female} \\ \text{FATHER} & \textit{person} \\ \text{MOTHER} & \textit{person} \\ \text{CHILDREN} & \textit{list of person} \\ \textit{female person} & \end{bmatrix}$$

At this point, we could ask ourselves if we really need the feature GENDER. The necessary information is already represented in the type *woman*. The question if specific information is represented by special features or whether it is stored in a type without a corresponding individual feature will surface again in the discussion of linguistic

analyses. Both alternatives differ mostly in the fact that the information which is modelled by types is not immediately accessible for structure sharing, which is discussed in Section 6.4.

Type hierarchies play an important role in capturing linguistic generalizations, which is why type hierarchies and the inheritance of constraints and information will be explained with reference to a further example in what follows. One can think of type hierarchies as an effective way of organizing information. In an encyclopedia, the individual entries are linked in such a way that the entries for monkey and mouse will each contain a pointer to mammal. The description found under mammal does therefore not have to be repeated for the subordinate concepts. In the same way, if one wishes to describe various electric appliances, one can use the hierarchy in Figure 6.1. The most

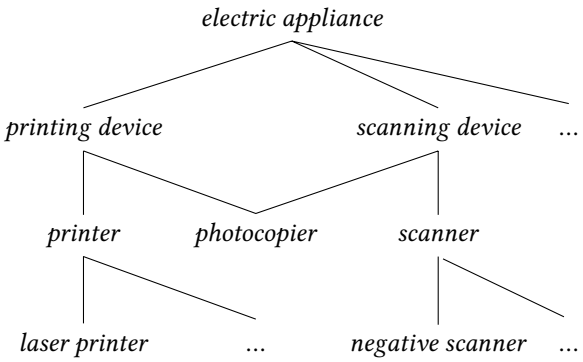

Figure 6.1: Non-linguistic example of multiple inheritance

general type *electrical device* is the highest in Figure 6.1. Electrical devices have certain properties, e. g. a power supply with a certain power consumption. All subtypes of *electrical device* "inherit" this property. In this way, *printing device* and *scanning device* also have a power supply with a specific power consumption. A *printing device* can produce information and a *scanning device* can read in information. A *photocopier* can both produce information and read it. Photocopiers have both the properties of scanning and printing devices. This is expressed by the connection between the two superordinate types and *photocopier* in Figure 6.1. If a type is at the same time the subtype of several superordinate types, then we speak of *multiple inheritance*. If devices can print, but not scan, they are of type *printer*. This type can have further more specific subtypes, which in turn may have particular properties, e. g. *laser printer*. New features can be added to subtypes, but it is also possible to make values of inherited features more specific. For example, the material that can be scanned with a *negative scanner* is far more restricted than that of the supertype *scanner*, since negative scanners can only scan negatives.

The objects that are modeled always have a maximally specific type. In the example above, this means that we can have objects of the type *laser printer* and *negative scanner* but not of the type *printing device*. This is due to the fact that *printing device* is not

maximally specific since this type has two subtypes.

Type hierarchies with multiple inheritance are an important means for expressing linguistic generalizations (Flickinger, Pollard & Wasow 1985; Flickinger 1987; Sag 1997). Types of words or phrases which occur at the very top of these hierarchies correspond to constraints on linguistic objects, which are valid for linguistic objects in all languages. Subtypes of such general types can be specific to certain languages or language classes.

6.3 Disjunction

Disjunctions can be used if one wishes to express the fact that a particular object can have various different properties. If one were to organize a class reunion twenty years after leaving school and could not recall the exact names of some former classmates, it would be possible to search the web for "Julia (Warbanow or Barbanow)". In feature descriptions, this "or" is expressed by a '$\vee$'.

(12)
$$\begin{bmatrix} \textsc{firstname} & \textit{julia} \\ \textsc{lastname} & \textit{warbanow} \vee \textit{barbanow} \\ \textit{person} & \end{bmatrix}$$

Some internet search engines do not allow for searches with 'or'. In these cases, one has to carry out two distinct search operations: one for "Julia Warbanow" and then another for "Julia Barbanow". This corresponds to the two following disjunctively connected descriptions:

(13)
$$\begin{bmatrix} \textsc{firstname} & \textit{julia} \\ \textsc{lastname} & \textit{warbanow} \\ \textit{person} & \end{bmatrix} \vee \begin{bmatrix} \textsc{firstname} & \textit{julia} \\ \textsc{lastname} & \textit{barbanow} \\ \textit{person} & \end{bmatrix}$$

Since we have type hierarchies as a means of expression, we can sometimes do without disjunctive specification of values and instead state the supertype: for *printer* $\vee$ *photocopier*, one can simply write *printing device* if one assumes the type hierarchy in Figure 6.1 on the preceding page.

6.4 Structure sharing

Structure sharing is an important part of the formalism. It serves to express the notion that certain parts of a structure are identical. A linguistic example for the identity of values is agreement. In sentences such as (14), the number value of the noun phrase has to be identical to that of the verb:

(14) a. Der Mann schläft.
the man sleeps
'The man is sleeping.'

b. Die Männer schlafen.
the men sleep
'The men are sleeping.'

c. * Der Mann schlafen.
the man sleep
Intended: 'The man are sleeping.'

The identity of values is indicated by boxes containing numbers. The boxes can also be viewed as variables.

When describing objects we can make claims about equal values or claims about identical values. A claim about the identity of values is stronger. Let us take the following feature description containing information about the children that Max's father and mother have as an example:

(15)
$$\begin{bmatrix}
\textsc{firstname} & \textit{max} \\
\textsc{lastname} & \textit{meier} \\
\textsc{date-of-birth} & \textit{10.10.1985} \\
\textsc{father} & \begin{bmatrix}
\textsc{firstname} & \textit{peter} \\
\textsc{lastname} & \textit{meier} \\
\textsc{children} & \left\langle \begin{bmatrix} \textsc{firstname} & \textit{klaus} \\ \textit{person} & \end{bmatrix}, \ldots \right\rangle \\
\textit{person} &
\end{bmatrix} \\
\textsc{mother} & \begin{bmatrix}
\textsc{firstname} & \textit{anna} \\
\textsc{lastname} & \textit{meier} \\
\textsc{children} & \left\langle \begin{bmatrix} \textsc{firstname} & \textit{klaus} \\ \textit{person} & \end{bmatrix}, \ldots \right\rangle \\
\textit{person} &
\end{bmatrix} \\
\textit{person} &
\end{bmatrix}$$

Notice that under the paths FATHER|CHILDREN and MOTHER|CHILDREN, we find a list of a description of a person with the first name Klaus. The question of whether the feature description is of one or two children of Peter and Anna cannot be answered. It is certainly possible that we are dealing with two different children from previous partnerships who both happen to be called Klaus.

By using structure sharing, it is possible to specify the identity of the two values:

(16)
$$\begin{bmatrix}
\textsc{firstname} & \textit{max} \\
\textsc{lastname} & \textit{meier} \\
\textsc{date-of-birth} & \textit{10.10.1985} \\
\textsc{father} & \begin{bmatrix}
\textsc{firstname} & \textit{peter} \\
\textsc{lastname} & \textit{meier} \\
\textsc{children} & \left\langle \boxed{1} \begin{bmatrix} \textsc{firstname} & \textit{klaus} \\ \textit{person} & \end{bmatrix}, \ldots \right\rangle \\
\textit{person} &
\end{bmatrix} \\
\textsc{mother} & \begin{bmatrix}
\textsc{firstname} & \textit{anna} \\
\textsc{lastname} & \textit{meier} \\
\textsc{children} & \langle \boxed{1}, \ldots \rangle \\
\textit{person} &
\end{bmatrix} \\
\textit{person} &
\end{bmatrix}$$

In (16), Klaus is a single child that belongs to both parents. Everything inside the brackets which immediately follow $\boxed{1}$ is equally present in both positions. One can think of $\boxed{1}$ as a pointer or reference to a structure which has only been described once. One question still remains open: what about Max? Max is also a child of his parents and should therefore also occur in a list of the children of his parents. There are two points in (16) where there are three dots. These ellipsis marks stand for information about the other children of Peter and Anna Meier. Our world knowledge tells us that both of them must have the same child namely Max Meier himself. In the following section, we will see how this can be expressed in formal terms.

6.5 Cyclic structures

We have introduced structure sharing in order to be able to express the fact that Max's parents both have a son Klaus together. It would not be enough to list Max in the child-lists of his parents separately. We want to capture the fact that it is the same Max which appears in each of these lists and furthermore, we have to ensure that the child being described is identical to the entire object being described. Otherwise, the description would permit a situation where Max's parents could have a second child also called Max. The description given in (17) can capture all facts correctly.

(17)
$$\boxed{2}\begin{bmatrix}
\textsc{firstname} & \textit{max} \\
\textsc{lastname} & \textit{meier} \\
\textsc{date-of-birth} & \textit{10.10.1985} \\
\textsc{father} & \begin{bmatrix}
\textsc{firstname} & \textit{peter} \\
\textsc{lastname} & \textit{meier} \\
\textsc{children} & \left\langle \boxed{1}\begin{bmatrix}\textsc{firstname} & \textit{klaus} \\ \textit{person}\end{bmatrix}, \boxed{2} \right\rangle \\
\textit{person}
\end{bmatrix} \\
\textsc{mother} & \begin{bmatrix}
\textsc{firstname} & \textit{anna} \\
\textsc{lastname} & \textit{meier} \\
\textsc{kinder} & \langle \boxed{1}, \boxed{2} \rangle \\
\textit{person}
\end{bmatrix} \\
\textit{person}
\end{bmatrix}$$

Structures such as those described in (17) are called cyclic because one ends up going in a circle if one follows a particular path: e. g. the path FATHER|CHILDREN|...|FATHER|CHILDREN|...[2] can be potentially repeated an infinite number of times.

6.6 Unification

Grammatical rules are written exactly like lexical entries in HPSG and Construction Grammar and are done so with the help of feature descriptions. For a word or a larger phrasal entity to be usable as daughter in a phrase licensed by some grammatical rule, the word or phrase must have properties which are compatible with the description of the daughters in the grammatical rule. If this kind of compatibility exists, then we can say that the respective items are *unifiable*.[3] If one unifies two descriptions, the result is a description which contains information from both descriptions but no additional information.

The way unification works can be demonstrated with feature descriptions describing people. One can imagine that Bettina Kant goes to the private detective Max Müller and wants to find a specific person. Normally, those who go to a detective's office only come with a partial description of the person they are looking for, e. g. the gender, hair color or date of birth. Perhaps even the registration number of the car belonging to the person is known.

[2] The dots here stand for the path to [2] in the list which is the value of CHILDREN. See Exercise 3.

[3] The term *unification* should be used with care. It is only appropriate if certain assumptions with regard to the formal basis of linguistic theories are made. Informally, the term is often used in formalisms where unification is not technically defined. In HPSG, it mostly means that the constraints of two descriptions lead to a single description. What one wants to say here, intuitively, is that the objects described have to satisfy the constraints of both descriptions at the same time (*constraint satisfaction*). Since the term *unification* is so broadly-used, it will also be used in this section. The term will not play a role in the remaining discussions of theories with the exception of explicitly unification-based approaches. In contrast, the concept of constraint satisfaction presented here is very important for the comprehension of the following chapters.

It is then expected of the detective that he or she provides information fitting the description. If we are looking for a blonde female named Meier (18a), then we do not want to get descriptions of a male red-head (18b). The descriptions in (18) are incompatible and cannot be unified:

(18) a.
$$\begin{bmatrix} \textsc{lastname} & \textit{meier} \\ \textsc{gender} & \textit{female} \\ \textsc{haircolor} & \textit{blonde} \\ \textit{person} & \end{bmatrix}$$

b.
$$\begin{bmatrix} \textsc{lastname} & \textit{meier} \\ \textsc{gender} & \textit{male} \\ \textsc{haircolor} & \textit{red} \\ \textit{person} & \end{bmatrix}$$

The description in (19) would be a possible result for a search for a blonde, female individual called Meier:

(19)
$$\begin{bmatrix} \textsc{firstname} & \textit{katharina} \\ \textsc{lastname} & \textit{meier} \\ \textsc{gender} & \textit{female} \\ \textsc{date-of-birth} & \textit{15.10.1965} \\ \textsc{haircolor} & \textit{blonde} \\ \textit{person} & \end{bmatrix}$$

Katharina Meier could also have other properties unknown to the detective. The important thing is that the properties known to the detective match those that the client is looking for. Furthermore, it is important that the detective uses reliable information and does not make up any information about the sought object. The unification of the search in (18a) and the information accessible to the detective in (19) is in fact (19) and not (20), for example:

(20)
$$\begin{bmatrix} \textsc{firstname} & \textit{katharina} \\ \textsc{lastname} & \textit{meier} \\ \textsc{gender} & \textit{female} \\ \textsc{date-of-birth} & \textit{15.10.1965} \\ \textsc{haircolor} & \textit{blond} \\ \textsc{children} & \langle\rangle \\ \textit{person} & \end{bmatrix}$$

(20) contains information about children, which is neither contained in (18a) nor in (19). It could indeed be the case that Katharina Meier has no children, but there are perhaps several people called Katharina Meier with the otherwise same properties. With this invented information, we might exclude one or more possible candidates.

It is possible that our detective Max Müller does not have any information about hair color in his files. His files could contain the following information:

(21)
$$\begin{bmatrix} \textsc{firstname} & \textit{katharina} \\ \textsc{lastname} & \textit{meier} \\ \textsc{gender} & \textit{weiblich} \\ \textsc{date-of-birth} & \textit{15.10.1965} \\ \textit{person} & \end{bmatrix}$$

These data are compatible with the search criteria. If we were to unify the descriptions in (18a) and (21), we would get (19). If we assume that the detective has done all his work, then Bettina Kant now knows that the person she is looking for has the properties of her original search plus the newly discovered properties.

6.7 Phenomena, models and formal theories

In the previous sections, we introduced feature descriptions with types. These feature descriptions describe typed feature structures, which are models of observable linguistic structures. In the definitions of types, one determines which properties of linguistic objects should be described. The type hierarchy together with type definitions is also referred to as a *signature*. As a grammarian, one typically uses types in feature descriptions. These descriptions contain constraints which must hold for linguistic objects. If no constraints are given, all values that are compatible with the specification in the signature are possible values. For example, one can omit the case description of a linguistic object such as *Frau* 'woman' since *Frau* can – as shown in (22) – appear in all four cases:

(22) a. Die Frau schläft. (nominative)
the.NOM woman sleeps

b. Wir gedenken der Frau. (genitive)
we commemorate the.GEN woman

c. Er hilft der Frau. (dative)
he helps the.DAT woman

d. Er liebt die Frau. (accusative)
he loves the.ACC woman

In a given model, there are only fully specified representations, that is, the model contains four forms of *Frau*, each with a different case. For masculine nouns such as *Mann* 'man', one would have to say something about case in the description since the genitive-singular form *Mann-es* differs from other singular forms, which can be seen by adding *Mann* into the examples in (22). (23) shows the feature descriptions for *Frau* 'woman' and *Mann* 'man':

(23) a. Frau 'woman'
$$\begin{bmatrix} \textsc{gender} & \textit{fem} \end{bmatrix}$$

b. Mann 'man'

$$\begin{bmatrix} \textsc{gender} & \textit{mas} \\ \textsc{case} & \textit{nominative} \vee \textit{dative} \vee \textit{accusative} \end{bmatrix}$$

Unlike (23b), (23a) does not contain a case feature since we do not need to say anything special about case in the description of *Frau*. Since all nominal objects require a case feature, it becomes clear that the structures for *Frau* must actually also have a case feature. The value of the case feature is of the type *case*. *case* is a general type which subsumes the subtypes *nominative*, *genitive*, *dative* and *accusative*. Concrete linguistic objects always have exactly one of these maximally specified types as their case value. The feature structures belonging to (23) are given in Figure 6.2 and Figure 6.3.

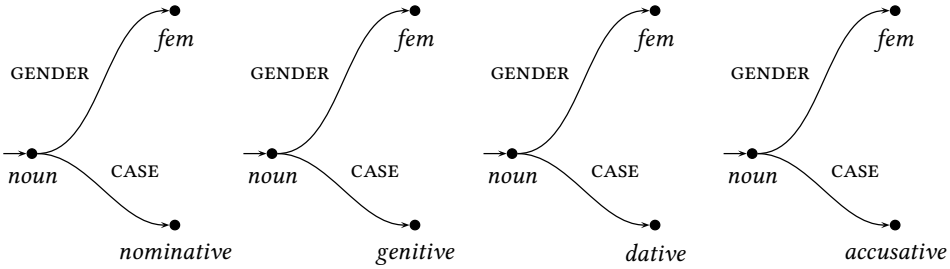

Figure 6.2: Feature structures for the description of *Frau* 'woman' in (23a)

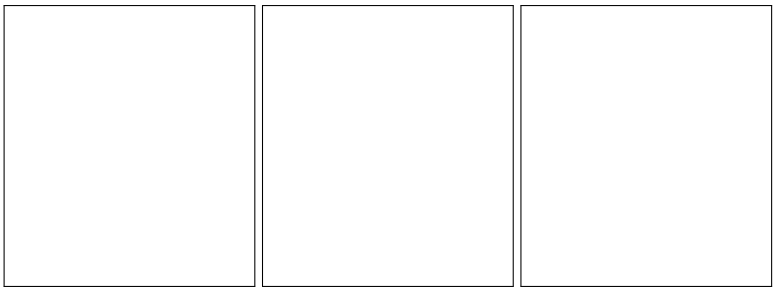

Figure 6.3: Feature structures for the description of *Mann* 'man' in (23b)

In these representations, each node has a certain type (*noun*, *fem*, *nominative*, ...) and the types in feature structures are always maximally specific, that is, they do not have any further subtypes. There is always an entry node (*noun* in the example above) and the other nodes are connected with arrows that are annotated with the feature labels (GENDER, CASE).

If we return to the example with people from the previous sections, we can capture the difference between a model and a description as follows: if we have a model of people that includes first name, last name, date of birth, gender and hair color, then it

follows that every object we model also has a birthday. We can, however, decide to omit these details from our descriptions if they do not play a role for stating constraints or formulating searches.

The connection between linguistic phenomena, the model and the formal theory is shown in Figure 6.4. The model is designed to model linguistic phenomena. Further-

Figure 6.4: Phenomenon, model and formal theory

more, it must be licensed by our theory. The theory determines the model and makes predictions with regard to possible phenomena.

Comprehension questions

1. What are the reasons for using types?
2. What is inheritance? What is special about multiple inheritance?
3. Are the following structures compatible, that is, can they be used to describe the same object?

(24)

$$\begin{bmatrix} \textsc{firstname} & \textit{max} \\ \textsc{lastname} & \textit{meier} \\ \textsc{father} & \begin{bmatrix} \textsc{firstname} & \textit{peter} \\ \textsc{lastname} & \textit{meier} \\ \textit{person} \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textsc{firstname} & \textit{max} \\ \textsc{lastname} & \textit{meier} \\ \textsc{father} & \begin{bmatrix} \textsc{firstname} & \textit{peter} \\ \textsc{lastname} & \textit{müller} \\ \textit{person} \end{bmatrix} \end{bmatrix}$$

(25)

$$\begin{bmatrix} \textsc{firstname} & \textit{max} \\ \textsc{lastname} & \textit{meier} \\ \textsc{father} & \begin{bmatrix} \textsc{firstname} & \textit{peter} \\ \textsc{lastname} & \textit{meier} \\ \textit{person} \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textsc{firstname} & \textit{max} \\ \textsc{lastname} & \textit{meier} \\ \textsc{mother} & \begin{bmatrix} \textsc{firstname} & \textit{ursula} \\ \textsc{lastname} & \textit{müller} \\ \textit{person} \end{bmatrix} \end{bmatrix}$$

Exercises

1. Think about how one could describe musical instruments using feature descriptions.

2. Come up with a type hierarchy for the word classes (*det*, *comp*, *noun*, *verb*, *adj*, *prep*). Think about the ways in which one can organize the type hierachy so that one can express the generalizations that where captured by the binary features in Table 3.1 on page 96.

3. In this chapter, we introduced lists. This may look like an extension of the formalism, but it is not as it is possible to convert the list notation into a notation which only requires feature-value pairs. Think about how one could do this.

4. (Additional exercise) The relation *append* will play a role in Chapter 9. This relation serves to combine two lists to form a third. Relational constraints such as *append* do in fact constitute an expansion of the formalism. Using relational constraints, it is possible to relate any number of feature values to other values, that is, one can write programs which compute a particular value depending on other values. This poses the question as to whether one needs such powerful descriptive tools in a linguistic theory and if we do allow them, what kind of complexity we afford them. A theory which can do without relational constraints should be preferred over one that uses relational constraints (see Müller (2007b: Chapter 20) for a comparison of theories).

   For the concatenation of lists, there is a possible implementation in feature structures without recourse to relational constraints. Find out how this can be done. Give your sources and document how you went about finding the solution.

Further reading

This chapter was designed to give the reader an easy-to-follow introduction to typed feature structures. The mathematical properties of the structures, type hierarchies and the combinatorial possibilities of such structures could not be discussed in detail here, but knowledge of at least part of these properties is important for work in computational linguistics and in developing one's own analyses. For more information, I refer the interested reader to the following publications: Shieber (1986) is a short introduction to the theory of Unification Grammar. There is a relatively general overview followed by the discussion of important grammar types such as DCG, LFG, GPSG, HPSG, PATR-II. Johnson (1988) describes the formalism of untyped feature structures in a mathematically precise way. Carpenter (1992) goes into the detail about the mathematical aspects of typed feature structures. The formalism developed by King (1999) for HPSG-grammars forms the basis for the formalism by Richter (2004), which currently counts as the standard formalism for HPSG.