Feature Matrices and Agreement: A Case Study for German Case

Remi van Trijp

Abstract

This paper illustrates the use of 'feature matrices', a technique for handling ambiguity and feature indeterminacy in feature structure grammars using unification as the single mechanism for processing. Both phenomena involve forms that can be mapped onto multiple, often conflicting values. This paper illustrates their respective challenges through German case agreement, which has become the litmus test for demonstrating how well a grammar formalism deals with multifunctionality. After reviewing two traditional solutions, the paper demonstrates how complex grammatical categories can be represented as feature matrices instead of single-valued features. Feature matrices allow a free flow of constraints on possible feature-values coming from any part of an utterance, and they postpone commitment to any particular value until sufficient constraints have been identified. All examples in this paper are operationalized in Fluid Construction Grammar, but the design principle can be extended to other unification-grammars as well.

1. Introduction

Natural languages are abundant with forms that can be mapped onto different, often conflicting feature-values. For example, the number of the English definite article *the* can either be singular or plural. Syntactic cues for arriving at the correct reading may come from various sources in an utterance:

(1) The **man** crossed the street.

(2) The fish **were** biting well that day.

(3) The antelope ran away when John tried to approach **them**.

In the first example, the Determiner Noun Construction identifies *the* as a singular definite article because it agrees in number with the singular noun form *man*. As illustrated in example (2), however, some nouns such as *fish* are underspecified for number themselves, so another source of information is required. Here, the plural verb form *were* offers the correct reading through subject-verb agreement. In the third sentence, neither the noun nor the verb are sufficient for finding out whether *the* is singular or plural; it is the pronoun *them* in the subordinate clause that achieves this. Examples such as (1–3) are instances of language *ambiguity*.

The literature on feature-based grammar formalisms distinguishes ambiguity from *feature indeterminacy* (also known as *feature neutrality*). Whereas ambiguous forms can only have one reading at the same time, indeterminate forms simultaneously satisfy two or more conflicting constraints (Dalrymple et al., 2009). For example, *sheep* is ambiguous because it cannot be singular and plural at the same time, as shown in (4). In the German example (5), on the other hand, the pronoun *was* 'what' is indeterminate because it simultaneously satisfies two conflicting constraints: it is assigned accusative case by the verb form *gegeben* 'given', and nominative by *ist* 'is'. Both examples are taken from Ingria (1990, p. 195 and 199).

(4) *The sheep that is ready are there.

(5)

Was	*du*	*mir*	*gegeben*	*hast,*	*ist*	*prächtig.*
what.N/A	you	me.D	given.A	have	is	wonderful.N

'What you have given to me is wonderful.'

Both phenomena pose great challenges on unification-based grammar formalisms in the following two ways:

1. Efficiency: Constraints on feature-values may come from many sources in the grammar. Moreover, these constraints may be propagated in any direction. In example (2), the verb propagates its value for its `number` feature to the subject, while the subject disambiguates the `person` feature of the verb. An adequate formalism therefore needs to allow a free flow of constraint propagation until a form can be disambiguated instead of trying to make choices already early on (at the high cost of computing unnecessary unifications).

2. Flexibility: Indeterminate features simultaneously satisfy conflicting constraints on their values, which seems to contradict the very nature of unification – a process that has to check the compatibility of two sources of information before combining them.

There are two traditional solutions for handling ambiguity and indeterminacy: disjunctive feature representation and type hierarchies (Copestake, 2002). Unfortunately, both techniques are problematic when it comes to more complex case studies. Disjunctive feature representation – typically favored by verbal approaches because of its elegant notation style – are highly inefficient in processing (Flickinger, 2000). Type hierarchies largely solve this problem of efficiency and are therefore more common in computational implementations. However, by using type hierarchies or other techniques as additional sources for checking the compatibility of features, grammatical analyses can grow needlessly complex without resolving all the linguistic issues.

This paper introduces an alternative design pattern that uses unification as its only processing mechanism. Instead of treating complex grammatical phenomena as single-valued features, they are represented in the form of 'feature matrices' that reflect particular grammatical paradigms and that use variables for indicating how specific forms fit into these paradigms. The technique of feature matrices is operationalized from the viewpoint that one of the main benefits of grammar is that it restricts the search space in processing (Steels & Wellens, 2006). The matrices avoid the inefficiency of disjunctive feature representation without resorting to complex, additional techniques of representation and processing. The technique is illustrated through German case agreement, whose ambiguity, agreement constraints and feature indetermination are notoriously difficult for the aforementioned traditional solutions. The solution can however easily be applied to other grammatical domains as well, as is shown elsewhere in this book for argument structure (van Trijp, 2011), verbal agreement (Beuls, 2011) and space (Spranger & Loetzsch, 2011). All examples in this paper have been operationalized in Fluid Construction Grammar, but the approach can be implemented in other unification-based formalisms as well.

2. Traditional Approaches to Ambiguity and Indeterminacy

This section illustrates and reviews two traditional solutions for handling ambiguity and indeterminacy in feature-based grammar formalisms using the German case system as an example. German articles, adjectives and nouns are marked for gender, number and case through morphological inflection. The system is notorious for its *syncretism* (i.e. the same form can be mapped onto different cells in the German case paradigm) and it can be considered as the litmus test for demonstrating whether a formalism adequately handles multifunctional categories. The paradigm of German definite articles is illustrated in Table 1.

Case	**SG-Masc**	**SG-Fem**	**SG-Neut**	**PL**
NOM	*der*	*die*	*das*	*die*
ACC	*den*	*die*	*das*	*die*
DAT	*dem*	*der*	*dem*	*den*
GEN	*des*	*der*	*des*	*der*

Table 1. *The morphological paradigm of German definite articles.*

2.1. Disjunctive feature representation

Case syncretism in German forms an interesting challenge for deep language processing formalisms because it interweaves three dimensions: case, number and gender. Disjunctive feature representation usually tries to represent this multifunctionality through listing the possibilities as disjunctions (i.e. separate alternatives). For example, the article *die* covers the nominative and accusative feminine singular case, or all plural nominative and accusative nouns. The following feature structure (adopted from Karttunen, 1984, p. 30) shows feature-value pairs between square brackets; disjunctions are presented by enclosing the alternatives in curly brackets ({ }).

(6)
$$\left[\begin{array}{ll} \text{AGREEMENT} & \left\{\begin{array}{l} \left[\begin{array}{ll}\text{GENDER} & f\\ \text{NUM} & sg\end{array}\right] \\ \left[\begin{array}{ll}\text{NUM} & pl\end{array}\right]\end{array}\right\} \\ \text{CASE} & \left\{ nom\ acc \right\} \end{array}\right]$$

Up until the 1980s, disjunctive feature representation was disallowed by most grammar formalisms. The technique finally made its way to unification-based grammars for handling exactly the kind of linguistic phenomena such as German articles (Karttunen, 1984), and its descriptive elegance has made it the most widespread way of representing multifunctionality in verbal (i.e. non-computational) grammar formalisms ever since.

Despite its elegance, disjunctive feature representation is not without flaws. Crysmann (2005) argues that the grammarian is often forced to make arbitrary implementation decisions. For example, the German base noun *Computer* can be represented using 'disjunctive normal form' (7) or using more compact 'nested disjunctions' (8–9; but here again it is arbitrary which dimension is chosen as the outer or inner disjunction). These different solutions all represent the fact that *Computer* can be nominative, accusative or dative singular, or that it can be nominative, accusative or genitive plural (ibid., ex. 2–4, disjunctions are here represented by $\vee$):

(7) $$\begin{bmatrix}\text{CASE} & \textit{nom}\\ \text{NUM} & \textit{sg}\end{bmatrix} \vee \begin{bmatrix}\text{CASE} & \textit{dat}\\ \text{NUM} & \textit{sg}\end{bmatrix} \vee \begin{bmatrix}\text{CASE} & \textit{acc}\\ \text{NUM} & \textit{sg}\end{bmatrix} \vee \begin{bmatrix}\text{CASE} & \textit{nom}\\ \text{NUM} & \textit{pl}\end{bmatrix} \vee \begin{bmatrix}\text{CASE} & \textit{gen}\\ \text{NUM} & \textit{pl}\end{bmatrix} \vee \begin{bmatrix}\text{CASE} & \textit{acc}\\ \text{NUM} & \textit{pl}\end{bmatrix}$$

(8) $$\begin{bmatrix}\text{CASE} & \textit{nom} \vee \textit{dat} \vee \textit{acc}\\ \text{NUM} & \textit{sg}\end{bmatrix} \vee \begin{bmatrix}\text{CASE} & \textit{nom} \vee \textit{gen} \vee \textit{acc}\\ \text{NUM} & \textit{pl}\end{bmatrix}$$

(9) $$\begin{bmatrix}\text{CASE} & \textit{nom} \vee \textit{acc}\end{bmatrix} \vee \begin{bmatrix}\text{CASE} & \textit{dat}\\ \text{NUM} & \textit{sg}\end{bmatrix} \vee \begin{bmatrix}\text{CASE} & \textit{gen}\\ \text{NUM} & \textit{pl}\end{bmatrix}$$

The above three solutions can all be successfully used to represent the German base noun *Computer*. However, it is a well-established fact that disjunctions are computationally very expensive (Flickinger, 2000). In fact, general unification of disjunctive features is NP-complete (Ramsay, 1990). Many studies therefore try to optimize processing of disjunctions through heuristics or approximation algorithms (e.g. Carter, 1990; Ramsay, 1990) or to eliminate disjunctions altogether whenever possible (Flickinger, 2000; Crysmann, 2005).

Figure 1 illustrates the problem. The Figure shows the search tree for parsing the utterance *Die Kinder gaben der Lehrerin die Zeichnung* 'the children gave the drawing to the (female) teacher'. The example uses a mini-grammar for German that consists of only six lexical entries: the definite articles *die* and *der*, the nouns *Kinder* 'children', *Lehrerin* 'female teacher' and *Zeichnung* 'drawing' and the verb form *gaben* 'gave.PL'. All lexical entries use disjunctive feature representation for their agreement features `case`, `gender` and `num` similar to the examples 6–9 above. Additionally, the grammar contains a Determiner-Noun construction that imposes agreement between the determiner and its head noun, and a ditransitive construction that captures the argument structure of the utterance.

What happens is that the disjunctions cause a split in the search tree whenever there are multiple alternatives possible. For example, *die Kinder* could be nominative or accusative plural, *der Frau* could be dative or genitive singular, and *die*

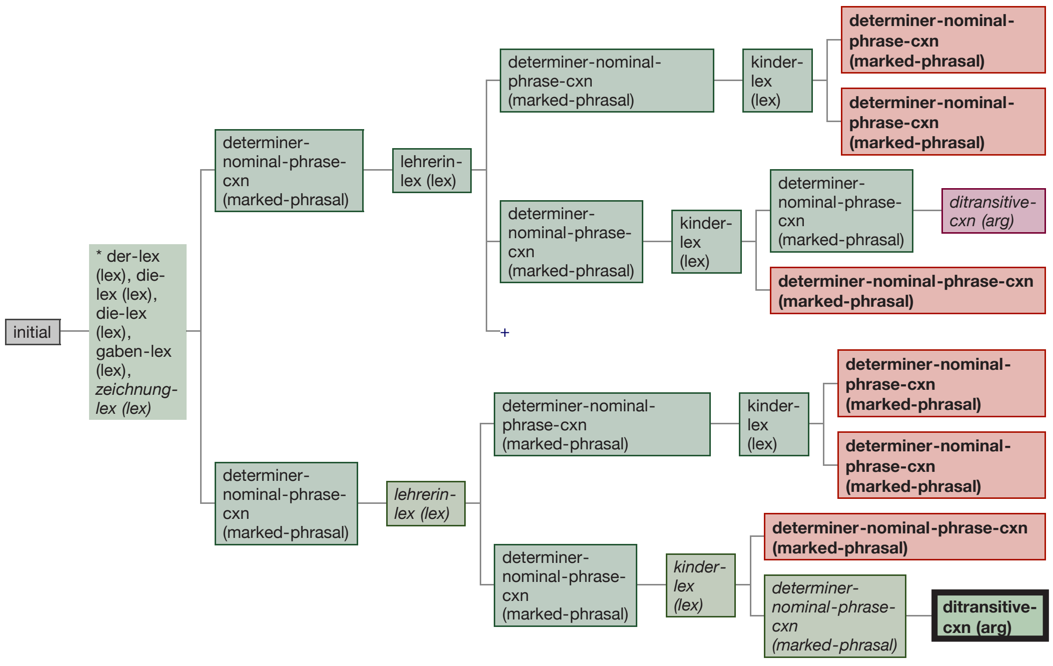

initial
* der-lex (lex), die-lex (lex), die-lex (lex), gaben-lex (lex), zeichnung-lex (lex)
determiner-nominal-phrase-cxn (marked-phrasal)
lehrerin-lex (lex)
kinder-lex (lex)
ditransitive-cxn (arg)
+

Figure 1. *Parsing of the utterance* Die Kinder gaben der Lehrerin die Zeichnung *'The children gave the drawing to the (female) teacher.' As can be seen, disjunctions force splits in the search tree regardless of syntactic context.*

Zeichnung could be nominative or accusative singular. This means that the search engine potentially has to consider seven false parses before the correct one is found. Additionally, every time a branch splits, the search space balloons accordingly because the search algorithm has to consider alternative orderings of applying the same constructions. The pluses in the Figure stand for these alternative branches that lead to duplicate nodes in the search tree. Their full expansion is not shown because of space limitations, but it should be obvious that detecting and pruning such duplicate nodes is a costly matter in terms of processing effort.

These efficiency issues also suggest that this search process is implausible from a psycholinguistic point of view because the example utterance is unambiguous for German speakers: *die Kinder* is the only candidate for being the subject because it is the only noun phrase that agrees with the main verb. This leaves only the

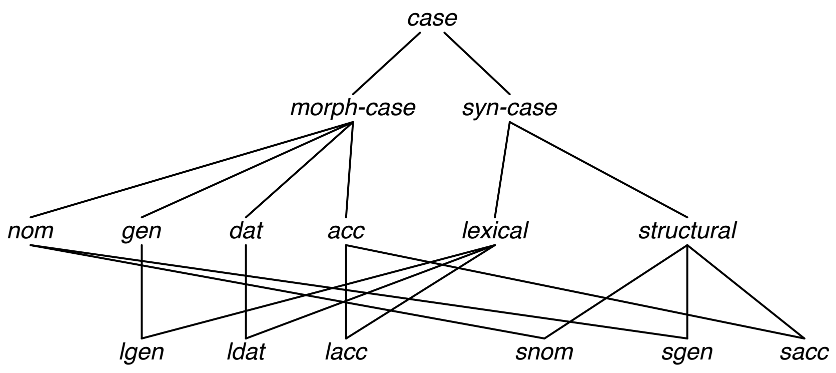


Figure 2. *A type hierarchy proposed for German case agreement (Heinz & Matiasek 1994, Figure adopted from Müller 2001).*

accusative slot open for *die Zeichnung*, and finally, *der Lehrerin* is unambiguously assigned dative case by the verb. In other words, the search tree does not reflect the processing choices that a natural language user would make as well, and they cause ambiguities even when the syntactic context is clear for native speakers.

2.2. Type hierarchies

Type hierarchies have taken up a central position in most contemporary grammar formalisms as an addition to the basic operation of unification. Such grammar formalisms, which are often called typed feature structure grammars, classify linguistic items in terms of types, which themselves are usually organized in a multiple inheritance network. (See Figure 2) For each type, particular constraints ('type constraints') can be defined, and each type has to satisfy the type constraints of all of its supertypes plus every constraint imposed on the type itself. A formalism's type system thus "acts as the defining framework for the rest of the grammar. For instance, it determines which structures are mutually compatible and which features can occur, and it sets up an inheritance system which allows generalizations to be expressed" (Copestake, 2002, p. 35). Even though type hierarchies do not exclude the use of disjunctions, they have sometimes been presented as a way to eliminate disjunctions whenever possible because they significantly increase efficiency (Flickinger, 2000). For German as well, various type hierarchies have been proposed (Heinz & Matiasek, 1994; Daniels, 2001; Müller, 2001).

2.2.1. *Problematic Agreement Constraints*

The type hierarchies that have been proposed for German case are 'combined' type hierarchies because they combine the three dimensions of case, number and gender. However, the German language poses serious challenges on such a combined hierarchy in, for example, coordinate constructions, which demand agreement of case among its conjuncts, but not of gender or number (Crysmann, 2005, ex. 10):

(10)

Ich	*helfe*	*der*	*und*	*dem*	*Mann.*
I	help	the.D.S.F	and	the.D.S.M	man

'I help this one and the man.'

In example (10), *dem Mann* and the pronominal *der* share the dative case, but they differ in gender, which is not possible in type hierarchies that use a single feature, because structure sharing in this approach enforces types to agree in number and gender as well (Müller, 2001). Solutions vary from introducing additional features (ibid.) to positing relational constraints (Daniels, 2001), but all of them return at least partially to disjunctive feature representation and therefore undo the efficiency gain of type hierarchies (Crysmann, 2005).

Crysmann proposes yet another solution in which he uses typed lists that are capable of abstracting away from particular dimensions if necessary. The solution therefore relies on two type hierarchies: the combined case-number-gender hierarchy and a hierarchy of case list types (as shown in Figure 3). There is a type constraint on the case list that restricts the first value of the list to the case agreement type of the combined case/number/gender type hierarchy. For instance, the type *nda-n-g* (for nominative, dative and accusative forms) may restrict the value of its `case` feature to an appropriate list type (ibid., ex. 12):

(11) $\textit{nda-n-g} \rightarrow \begin{bmatrix}\text{CASE} & \textit{nda-list}\end{bmatrix}$

Even though this solution works, it is clear that it requires quite a complex architecture only for isolating the relevant dimensions of the type hierarchy.

2.2.2. *The Problem of Feature Indeterminacy*

A second problem for type hierarchies is feature indetermination, as illustrated in (12) (Pullum & Zwicky, 1986; quoted from Crysmann, 2005, p. 24):

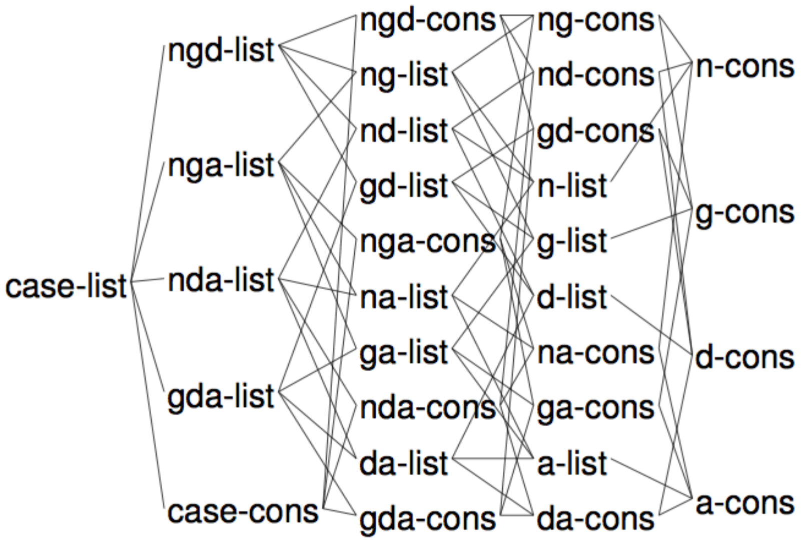


Figure 3. *In order to capture agreement constraints in German coordination structures, Crysmann (2005, ex. 11) proposes a hierarchy of case list types besides the combined type hierarchy of case, number and gender. The hierarchy goes from from super- to subtypes (left to right).*

(12) *Er findet und hilft Frauen.*
he finds.A and helps.D women.A/D

‘He finds and helps women.’

The verb *finden* ‘to find’ normally takes an accusative complement, whereas *helfen* ‘to help’ takes a dative complement. *Frauen* is underspecified and can be both accusative or dative. A sentence such as **Er findet und hilft Kindern* ‘He finds and helps children’, on the other hand, is ungrammatical because *Kindern* can only be dative and hence clashes with the requirements of the verb *finden*. Based on such examples, it has been argued by Ingria (1990) that unification is not the best technique for syntactic agreement and case assignment, and that compatibility checks are needed instead.

People have gone to great lengths to counter Ingria’s claim, especially within the HPSG framework (Müller, 1999; Daniels, 2001; Sag, 2003). One solution is to augment the type hierarchy to explicitly contain neutral (or indeterminate) types (Levine et al., 2001) that can act as if they have multiple values. In example (12), the word *Frauen* would have a neutral feature so it may act as though it has both dative and accusative feature values.

Unfortunately, it is very hard to decide when types should be treated as neutral (i.e. indetermined) or ambiguous. Moreover, as argued by Crysmann (2005), such a solution leads to drastic increases in the amount of lexical ambiguity. Crysmann writes that the apparent incompatibility of feature indetermination and underspecification cannot be overcome using a single type hierarchy. Instead, he proposes two partially independent hierarchies, one for ambiguity or internal case (*i-case*), and one for indetermination (*e-case*, see Figure 4).

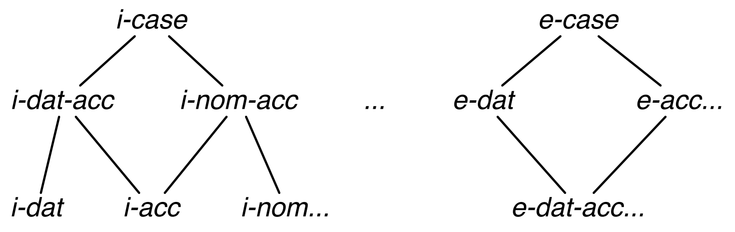


Figure 4. *Two partially independent hierarchies have been proposed for solving feature indetermination using typed feature structures. (Figure adopted from Crysmann 2005, ex. 34.)*

Roughly speaking, more specific types in one hierarchy will be compatible with less specific types in the other, and vice versa. (see Crysmann 2005 for the detailed technical discussion.) For example, the ambiguous form *Frauen* has the value *i-dat-acc*. This underspecified internal case unifies with overspecified *e-dat-acc*, which itself was obtained by coordinating the verbs *finden* and *helfen* that subcategorize for an *e-acc* and *e-dat* complement respectively. The specific value *i-dat* for *Kindern*, however, does not unify with overspecified *e-dat-acc*. Crysmann thus offers a working solution that keeps the efficiency of type hierarchies. However, the complexity of the approach also raises the question whether it stretches the limits of unification and typed feature structures too far, and whether other techniques might be more suited, as already suggested by Ingria (1990).

3. Feature Matrices

The alternative proposal presented in this paper is to represent complex grammatical categories as 'feature matrices'. This solution is inspired by 'distinctive features' in phonology that are used for classifying sounds in terms of binary values such as [voiced +] for /d/ and [voiced -] for /t/. We can easily extrapolate this idea to grammar and treat grammatical paradigms in terms of relevant distinctions.

How can we capture relevant distinctions for German case? Assume that case is not a feature with a single value, but an array of the case paradigm of that language. Each case is explicitly represented as a feature whose value can be '+' or '–', or left unspecified through a variable (indicated by a question mark).

3.1. Exploiting Underspecification

Returning to the example *Die Kinder gaben der Lehrerin die Zeichnung* 'the children gave the drawing to the teacher' (section 2.1) and ignoring genitive for the time being, the `case` feature of the definite article *die* and the noun *Zeichnung* could be represented as follows:

(13) die:
$$\begin{bmatrix} \text{CASE} & \begin{bmatrix} \text{nom} & \mathit{?nom} \\ \text{acc} & \mathit{?acc} \\ \text{dat} & - \end{bmatrix} \end{bmatrix}$$

(14) Zeichnung:
$$\begin{bmatrix} \text{CASE} & \begin{bmatrix} \text{nom} & \mathit{?nom} \\ \text{acc} & \mathit{?acc} \\ \text{dat} & \mathit{?dat} \end{bmatrix} \end{bmatrix}$$

The above representation, which is a simplification for illustration purposes only, captures the fact that *die* is ambiguous for nominative and accusative, but that it excludes dative. *Zeichnung* can be assigned any of these three cases.

Remember from Figure 1 that disjunctive feature representation forces a split in the search tree between a nominative and accusative reading of *die Zeichnung*, even though the syntactic context is unambiguous. Feature matrices avoid this problem because they make use of underspecification. Unifying *die* and *Zeichnung* leads to the following feature matrix, which can still be assigned nominative or accusative case later on, but which already excludes dative:

(15) die Zeichnung:
$$\begin{bmatrix} \text{CASE} & \begin{bmatrix} \text{nom} & \mathit{?nom} \\ \text{acc} & \mathit{?acc} \\ \text{dat} & - \end{bmatrix} \end{bmatrix}$$

3.2. Empowering the Matrix through Variables

This section shows how unification can efficiently handle complex grammatical categories such as German case agreement without resorting to additional techniques, if these categories are represented as feature matrices. This section also shows that feature matrices can elegantly solve tricky cases such as identity constraints in coordination and feature indetermination without positing additional constraints on processing or feature types. In reconsidering Table 1, this time we replace every cell in the table by a variable. This leads to the feature matrix for German case that is shown in Table 2.

Case	**S-M**	**S-F**	**S-N**	**PL**
?NOM	*?nom-s-m*	*?nom-s-f*	*?nom-s-n*	*?nom-pl*
?ACC	*?acc-s-m*	*?acc-s-f*	*?acc-s-n*	*?acc-pl*
?DAT	*?dat-s-m*	*?dat-s-f*	*?dat-s-n*	*?dat-pl*
?GEN	*?gen-s-m*	*?gen-s-f*	*?gen-s-n*	*?gen-pl*

Table 2. *The feature matrix for German case.*

Each cell in this matrix represents a specific feature bundle that combines the features case, number, and person. For example, the variable `?nom-s-m` stands for ‘nominative singular masculine’. Since plural forms do not mark differences in gender, only one plural cell is included for each case. Note that also the cases themselves have their own variable (`?nom`, `?acc`, `?dat` and `?gen`). As illustrated later, this column allows us to single out a specific dimension of the matrix for constructions that only care about case distinctions but abstract away from gender or number. Moreover, this additional column of variables captures crucial correlations between the various alternatives of case-gender-number assignment.

Each linguistic item fills in as much information as possible in this case matrix. For example, the definite article *der* underspecifies its potential values and rules out all other options through ‘–’, as shown in Table 3.

Case	**S-M**	**S-F**	**S-N**	**PL**
?nom-s-m	*?nom-s-m*	–	–	–
–	–	–	–	–
?dat-s-f	–	*?dat-s-f*	–	–
?gen	–	*?gen-s-f*	–	*?gen-pl*

Table 3. *The feature matrix for* der.

Note that the variable name for the nominative case `?nom-s-m` is the same as the one for the cell of nominative-singular-masculine, which means that if the article unifies with a masculine noun, it is automatically disambiguated as a nominative article, and vice versa, if the article is assigned nominative case, we can infer that it is masculine. The same goes for the dative case.

The string *Lehrerin* ‘teacher.F.SG’ rules out all plural forms but allows any case assignment. Since this noun is feminine, the single-dimension variables for case are the same ones as those that fill the singular-feminine cells in the matrix, as shown in Table 4.

Case	S-M	S-F	S-N	PL
?nom-s-f	–	*?nom-s-f*	–	–
?acc-s-f	–	*?acc-s-f*	–	–
?dat-s-f	–	*?dat-s-f*	–	–
?gen-s-f	–	*?gen-s-f*	–	–

Table 4. *The feature matrix for* Lehrerin.

Unification of *der* and *Lehrerin* only leaves the cells for dative and genitive feminine-singular open. In other words, *der Lehrerin* can only fill a dative or genitive slot. Other constructions may then later assign a ‘+’ value to one of the two cases. The resulting feature matrix is shown in Table 5.

Case	S-M	S-F	S-N	PL
–	–	–	–	–
–	–	–	–	–
?dat-s-f	–	*?dat-s-f*	–	–
?gen-s-f	–	*?gen-s-f*	–	–

Table 5. *The feature matrix for* der Leherin.

The efficiency of this technique is illustrated in Figure 5, which shows the search tree for parsing the same utterance *Die Kinder gaben der Lehrerin die Zeichnung* using feature matrices in the grammar. As opposed to the search with disjunctions (see Figure 1), feature matrices do not cause splits in the search tree unless there is an actual ambiguity in the language. Instead, they postpone commitment to any particular value as long as possible and thus allow information and constraints to be filled in by every part of the linguistic inventory.

Besides the enormous efficiency gain and a more plausible search process, feature matrices only require unification as the standard processing mechanism without additional sources for checking compatibility of information. The technique, therefore, seems to be a very elegant solution for representing and processing multifunctional categories. The real question, however, is whether feature matrices are also expressive enough to deal with those cases where traditional solutions are struggling. I claim that the answer is yes, and I will demonstrate why in the following two sections.

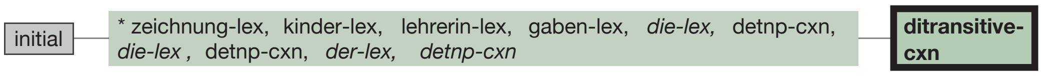

initial
* zeichnung-lex, kinder-lex, lehrerin-lex, gaben-lex, die-lex, detnp-cxn, die-lex , detnp-cxn, der-lex, detnp-cxn
ditransitive-cxn

Figure 5. *The search tree for* Die Kinder gaben der Lehrerin die Zeichnung *using feature matrices in the grammar.*

4. Disambiguation in Coordination Constructions

I will first return to the challenge of likeness constraints in coordination in German. For the sake of convenience, example 10 is repeated:

(16)	*Ich*	*helfe*	*der*	*und*	*dem*	*Mann.*
	I	help	the.D.S.F	and	the.D.S.M	man

‘I help this one and the man.’

The challenge here is whether feature matrices can impose an agreement constraint on the case values of *der* and *dem Mann*, but not on their number or gender. Additionally, the ambiguous pronoun *der* should be disambiguated as dative-feminine-singular in this syntactic context. Finally, the phrase as a whole should be plural. As it turns out, the solution is rather straightforward if we again exploit the expressive power of variables.

Evidence for the correct case value comes from two sources in this example: first of all, the verb *helfen* ‘to help’ always takes a dative complement. Secondly, the noun phrase *dem Mann* ‘the man’ is unambiguously dative-singular-masculine. If we unify the case matrix for *dem* and *Mann* in the same way as illustrated in the previous section, we get Table 6.

The case matrix for *der* was already shown in Table 3. All we need now is a coordination construction that imposes case agreement by simply repeating the

Case	**S-M**	**S-F**	**S-N**	**PL**
–	–	–	–	–
–	–	–	–	–
?dat-s-m	*?dat-s-m*	–	–	–
–	–	–	–	–

Table 6. *The feature matrix for* dem Mann.

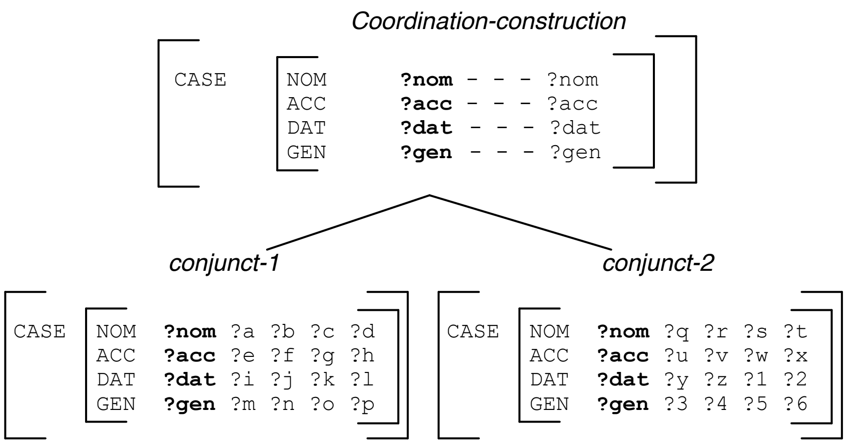


Figure 6. *This coordination construction shares variables with its conjuncts for the single dimension of case, but not for the feature bundles that cut across the other dimensions for number and gender. The coordinated phrase as a whole is plural even though its conjuncts may be singular.*

variables for the single dimension of case (the first column in the matrix) in the two conjuncts and in a third case matrix for the overarching coordination structure. This is shown in Figure 6. Note also that the matrix of the coordination structure has the same variables for its single dimension of case (first column) as for its plural cells (last column). This means that the whole coordination phrase has a plural value, even though its conjuncts may be singular.

By repeating the variables `?nom`, `?acc`, `?dat` and `?gen` in all three matrices, nominative, accusative and genitive are ruled out because these three cases were already assigned '–' by the matrix of *dem Mann*. Since only the single-dimension variables were shared by the three matrices, the matrix of the coordination allows its conjuncts to be of any number or gender as long as they are dative. The phrase as a whole can now only be dative-plural, as shown in Table 7.

Case	S-M	S-F	S-N	PL
–	–	–	–	–
–	–	–	–	–
?dat-pl	–	–	–	*?dat-pl*
–	–	–	–	–

Table 7. *The feature matrix for the coordinated structure after applying the coordination construction. (see Figure 6.)*

This means that the first and third parts of the challenge of agreement constraints in coordination have been successfully addressed. The second challenge was that *der* had to be disambiguated as a feminine pronoun. This is successful as well, as can be seen in Table 8, which shows the matrix of *der* after application of the coordination rule.

Case	S-M	S-F	S-N	PL
–	–	–	–	–
–	–	–	–	–
?dat-s-f	–	*?dat-s-f*	–	–
–	–	–	–	–

Table 8. *The feature matrix for* der *after application of the coordination construction (Figure 6).*

In sum, feature matrices provide a straightforward way of implementing agreement constraints that does not require any additional data structures such as typed lists and their corresponding hierarchies. At the same time they exploit underspecification to the fullest, and thereby postpone commitment to any particular value until needed.

5. Handling Feature Indetermination

Besides agreement constraints, it is straightforward to solve feature indetermination as well. For convenience's sake, example 12 is repeated here, as well as an ungrammatical example of the same construction:

(17)

Er	*findet*	*und*	*hilft*	*Frauen.*
he	finds.A	and	helps.D	women.A/D

‘He finds and helps women.’

(18)

**Er*	*findet*	*und*	*hilft*	*Kinder.*
he	finds.A	and	helps.D	children.A

‘He finds and helps children.’

These constructions can be accounted for without resorting to disjunctions, introducing neutral types or using separate type hierarchies for feature indetermination and ambiguity. Assume that German verbs have a valence feature through which they assign case to other phrases in an utterance. The verb *finden* ‘to find’ is a transitive verb that takes an accusative complement, which can be represented as follows (genitive is ignored here for illustration purposes):

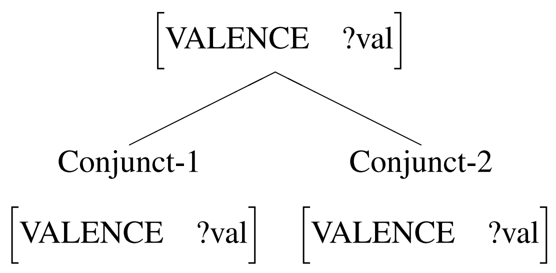
(19)
$$\begin{bmatrix} \text{VALENCE} & \begin{bmatrix} \text{subject} & \begin{bmatrix} \text{nom} & + \\ \text{acc} & - \\ \text{dat} & - \end{bmatrix} \\ \text{object} & \begin{bmatrix} \text{nom} & - \\ \text{acc} & + \\ \text{dat} & ?d \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The feature `valence` states that the verb may occur with a subject and object. Each grammatical function contains a case matrix, which is here reduced to a single dimension for illustrative purposes. The matrix for subject states that it has to be nominative (and nothing else). The object cannot be nominative but rather must be accusative. The value of the feature `valence` of the verb *helfen* ‘to help’ looks exactly the same except for the difference that *helfen* assigns the dative case to its object:

(20)
$$\begin{bmatrix} \text{VALENCE} & \begin{bmatrix} \text{subject} & \begin{bmatrix} \text{nom} & + \\ \text{acc} & - \\ \text{dat} & - \end{bmatrix} \\ \text{object} & \begin{bmatrix} \text{nom} & - \\ \text{acc} & ?a \\ \text{dat} & + \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Note that the matrices of the objects for both verbs contain a variable, which means that even though each verb assigns a particular case to its complement, it does not completely rule out either accusative or dative case. The reason for this variable is that if two verbs are coordinated, their valencies need to be able to unify with each other, which can, again, be achieved through the equality of variables:

(21) Verbal coordination construction



By repeating the variable `?val` as the value of the `valence` feature of both conjuncts, we state that the values of these features have to unify with each other. By repeating the variable in the overarching coordination construction as well, the unit as a whole will have this unified valence as the value of its own `valence` feature. Unification of the valencies of both verbs results in the following feature structure:

(22)

$$\begin{bmatrix} \text{VALENCE} & \begin{bmatrix} \text{subject} & \begin{bmatrix} \text{nom} & + \\ \text{acc} & - \\ \text{dat} & - \end{bmatrix} \\ \text{object} & \begin{bmatrix} \text{nom} & - \\ \text{acc} & + \\ \text{dat} & + \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The feature matrix of the object of the verb thus assigns ‘+’ to both the accusative and dative case, which means that only objects that are underspecified for both cases can effectively satisfy the constraints. This is true for *Frauen* ‘women’, as is shown in Table 9. The string *Kinder*, however, excludes dative case (which would take the form *Kindern*), as shown in Table 10, explaining the ungrammaticality of example 18.

Case	**S-M**	**S-F**	**S-N**	**PL**
?nom-pl	–	–	–	*?nom-pl*
?acc-pl	–	–	–	*?acc-pl*
?dat-pl	–	–	–	*?dat-pl*
?gen-pl	–	–	–	*?gen-pl*

Table 9. *The string* Frauen *is underspecified for case.*

Case	**S-M**	**S-F**	**S-N**	**PL**
?nom-pl	–	–	–	*?nom-pl*
?acc-pl	–	–	–	*?acc-pl*
–	–	–	–	–
?gen-pl	–	–	–	*?gen-pl*

Table 10. *The string* Kinder *excludes a dative reading.*

Since the objective of this paper is to illustrate the use of feature matrices rather than offering a full account of German, the above approach assumes that only acc-dat indeterminacy occurs in German. However, the anonymous reviewers of this paper rightfully pointed out that German has examples of nom-acc and gen-dat as well. In that case, as already suggested by one of the reviewers, no minuses are needed in the matrix, hence the verb's task is to only assign positive values instead of ruling out hypotheses. The question then becomes whether those open variables make the matrices too permissive and thus harmful for processing. The answer is no, since a positive value '+' really is a commitment to a certain value, which means that only complements that are compatible with that value are allowed. So even with open variables, the feature matrix will allow utterances such as *Er findet Kinder* 'he finds children', but not **Er hilft Kinder* 'he helps children'. In sum, the matrices can also handle feature indetermination using simply unification instead of resorting to additional operations.

6. Implementing Feature Matrices in Fluid Construction Grammar

Since unification is a very general operation, most unification-based grammars have added various mechanisms to constrain the possible values of a feature, such as special atoms (e.g. Functional Unification Grammar; Kay, 1985), coherence and appropriateness conditions (e.g. Lexical-Functional Grammar; Bresnan, 1982), feature co-occurrence restrictions (e.g. Generalized Phrase Structure Grammar; Gazdar et al., 1985), and type hierarchies (Copestake, 2002). All these approaches try to formalize language in the same way a logical calculus can be formalized, and assume a finite list of features.

Fluid Construction Grammar, on the other hand, tries to capture the 'fluid' and 'living aspects' of language (Steels, 2011b). This is one of the reasons why the current FCG-system does not type or impose appropriateness conditions on its feature-value pairs: as languages change over time, the number of features and appropriate values for them may change as well.[1] Since feature matrices rely solely on unification as the mechanism for processing, they can easily be implemented in FCG in order to process multifunctional grammatical categories in an efficient way. It is straightforward to represent feature matrices in FCG using a bracketed notation. For example, the `case` feature of the word *Frauen* (see Table 9) looks as follows:

(23)
```
(case (==1 (nom ?nom-pl - - - ?nom-pl)
           (acc ?acc-pl - - - ?acc-pl)
           (dat ?dat-pl - - - ?dat-pl)
           (gen ?gen-pl - - - ?gen-pl)))
```

The special operator `==1` ensures that the specific cases (nominative, accusative, dative and genitive) may only appear once in the feature-value, but their order doesn't matter. Since the matrices do not expect any extensions of the FCG-interpreter, it is possible to include them directly into the definition of a construction. For example, using the `def-lex-cxn` template (Steels, 2011a), *Frauen* could be defined as follows:

1. In principle, typed hierarchies could model language change as well. However, every change in a hierarchy is non-local, meaning that even minor drifts in the language may have dramatic effects on the performance of the whole system.

(24)
```
(def-lex-cxn Frauen-lex
 (def-lex-skeleton Frauen-lex
  :meaning (== (women ?women-ref))
  :args (?women-ref)
  :string "Frauen")
 (def-lex-cat Frauen-lex
  :sem-cat (==1 (sem-function identifier)
                (is-animate? +))
  :syn-cat
   (==1 (syn-function nominal)
        (lex-cat noun)
        (case
         ((nom ?nom-pl - - - ?nom-pl)
          (acc ?acc-pl - - - ?acc-pl)
          (dat ?dat-pl - - - ?dat-pl)
          (gen ?gen-pl - - - ?gen-pl))))))
```

This representation style is also suited for indeterminate features. Indeterminate feature matrices can be recognized by the fact that there are either multiple conflicting cells with the value '+', or that there are still variables in cells that are in conflict with a cell that already has been assigned '+'. The following feature structure shows the valence of the verb form *findet* (also see example 19), which takes an accusative object:

(25)
```
(syn-valence
 (==1
  (subject
   ((filler-unit ?subject-unit)
    (case
     ((nom + ?nom-s-m ?nom-s-f ?nom-s-n -)
      (acc - - - - -)
      (dat - - - - -)
      (gen - - - - -)))))
  (object
   ((filler-unit ?object-unit)
    (case
     ((nom - - - - -)
      (acc + ?acc-s-m
           ?acc-s-f ?acc-s-n ?acc-pl)
      (dat ?dat ?dat-s-m
           ?dat-s-f ?dat-s-n ?dat-pl)
      (gen - - - - -)))))))
```

Readers who are familiar with the template `def-lex-valence` (see van Trijp, 2011) will notice that the syntactic roles `subject` and `object` in (25) take more complex values than provided by that template. This is necessary in order to effect complex agreement, a topic returned to below. For now it suffices to know that the above valence states that *findet* can take a subject and an object, and that the subject has to be nominative singular, whereas the object has to be accusative. The complete row of the dative is left unspecified, which allows the object to be indeterminate for accusative or dative.

Examples (24–25) show that despite their technical simplicity, feature matrices require careful design in how variable equalities are used. For complex grammatical categories, this approach soon becomes cumbersome and defining feature matrices for them by hand is an error-prone process. Therefore, the following sections offer some general templates for using feature matrices.

6.1. Grammatical Paradigm

The first important step is to identify and define the *paradigm* of a grammatical phenomenon. By doing so it becomes possible to define how a particular construction subscribes itself to that paradigm. For example, a language may have a three-way distinction between subject, direct object and indirect object in how it assigns grammatical roles in a sentence. The template `define-paradigm` allows us to define and store this linguistic observation:

(26)
```
(define-paradigm *english-grammatical-roles*
 :dimensions
    ((subject object indirect-object)))
```

This results in the following paradigm or matrix (the first element in each list is the label of the grammatical role, the second element stands for the particular value or dimension in the paradigm):

(27)
```
((subject subject)
 (direct-object direct-object)
 (indirect-object indirect-object))
```

Of course, the German case system is more complex than a simple three-way distinction and it cuts across three dimensions. The value of the template's slot `:dimensions` therefore contains two lists: a first one for defining the main dimension of the matrix (here: the four cases) and a second one for defining the other dimensions that should be combined with the main dimension (here: number and gender):

(28)
```
(define-paradigm *german-case*
 :dimensions ((nom acc dat gen)
              (s-m s-f s-n pl)))
```

When expanded, the German case paradigm looks as follows:

(29)
```
((nom nom nom-s-m nom-s-f nom-s-n nom-pl)
 (acc acc acc-s-m acc-s-f acc-s-n acc-pl)
 (dat dat dat-s-m dat-s-f dat-s-n dat-pl)
 (gen gen gen-s-m gen-s-f gen-s-n gen-pl))
```

The same template can be used for any feature that requires an array of values rather than a single value. Other examples in this volume can be found for Hungarian vowel harmony and agreement (Beuls, 2011) and spatial expressions (Spranger & Loetzsch, 2011). Feature matrices have also been applied to German field topology and information structure (Micelli, 2012).

6.2. Subscribing Ambiguous Constructions and Units to the Paradigm

After defining the grammatical paradigm of a feature, we can exploit this paradigm for defining the feature matrix of a specific construction. What is needed here is a general template for specifying (a) in which unit-feature a matrix should be added or replaced, (b) how a particular unit or construction fits in a given paradigm. The template `def-lex-feature-matrix` satisfies both requirements and is illustrated here for *Frauen*:

(30)
```
(def-lex-feature-matrix Frauen-lex
 :feature (:syn-cat :case)
 :dimensions (pl)
 :paradigm *german-case*)
```

The first slot :feature is used for specifying the location and the name of the feature matrix. In the above example, the feature matrix will be put into the lexical construction's syn-cat feature and be called case. The grammatical paradigm that the template must use is specified through the slot :paradigm. The slot :dimensions, then, lists all possible values that can be assigned to a linguistic form given a particular paradigm. Here, only pl (plural) is specified, hence the template looks into the *german-case* paradigm and verifies which cells are plural. This check is performed based on the name of the cell: nom-pl, acc-pl, dat-pl and gen-pl all contain the symbol pl hence these are the cells that are retained by the template as possible values.

The template then figures out whether it can already assign '+', '–' or a variable to the cells in the matrix. It also automatically checks whether there are dependencies among cells that can be represented through variable equalities. In the case of *Frauen*, the template creates such variable equalities for the main dimension of the matrix. The resulting feature matrix is the same one as shown in example (23).

One example where the template can already assign a '+' is the personal pronoun *wir* 'we', which is always nominative plural. Suppose we have a lexical construction for *wir*, we can define its case matrix as follows:

(31)
```
(def-lex-feature-matrix wir-lex
 :feature (:syn-cat :case)
 :dimensions (nom-pl)
 :paradigm *german-case*)
```

This definition results in the following feature matrix for case:

(32)
```
(case  (==1 (nom + - - - +)
            (acc - - - - -)
            (dat - - - - -)
            (gen - - - - -)))
```

The first person singular pronoun *ich* 'I' is always nominative-singular. Its feature matrix for case is defined as follows:

(33)
```
(def-lex-feature-matrix ich-lex
 :feature (:syn-cat :case)
 :dimensions (nom-s)
 :paradigm *german-case*)
```

The template will introduce three variables in the feature matrix for the three cells that are nominative-singular. On top of that, it automatically infers that the value for the main dimension `nom` has to be ‘+’:

(34)
```
(case
 (==1 (nom + ?nom-s-m ?nom-s-f ?nom-s-n -)
      (acc - - - - -)
      (dat - - - - -)
      (gen - - - - -)))
```

6.3. Subscribing Indeterminate Constructions and Units to the Paradigm

Even though technically speaking, indeterminacy is not handled differently than ambiguity by feature matrices, there is a conceptual difference from a linguistic point of view which translates itself into a separate keyword in the `def-lex-feature-matrix` template. In order to grasp the examples in this section, the reader is expected to be familiar with the approach to argument structure as explained in more detail by van Trijp (2011). In this approach, the German verb form *findet* ‘finds’ can be defined as follows:

(35)
```
(def-lex-cxn findet-lex
 (def-lex-skeleton findet-lex
  :meaning (== (find ?ev)
               (finder ?ev ?finder)
               (found ?ev ?found))
  :args (?ev)
  :string "findet")
 (def-lex-cat findet-lex
  :sem-cat (==1 (sem-function predicating))
  :syn-cat (==1 (syn-function verbal)
                (lex-cat verb)))
 (def-lex-valence findet-lex
  :sem-roles ((agent finder)
              (patient found))
  :syn-roles (subject object)))
```

Using this definition, the verb’s `syn-valence` looks as follows:

(36)
```
(syn-valence
  ((subject ((filler-unit ?subject-unit)))
   (object ((filler-unit ?object-unit)))))
```

So far, the `syn-valence` in (36) does not impose any further constraints on the units that fill the subject or object roles. The `def-lex-feature-matrix` template can now be used for assigning the nominative case to the subject in the same way as illustrated in the previous section:

(37)
```
(def-lex-feature-matrix findet-lex
 :feature
   (:syn-cat :syn-valence :subject :case)
 :dimensions (nom-s)
 :paradigm *german-case*)
```

As can be seen in the `:feature` slot, it is possible to specify that the `case` feature needs to be found in the `subject` feature, which itself is part of the value of the verb's `syn-valence` feature, which is in its turn located in the unit-feature `syn-cat`. Next, the verb's object needs to be assigned the accusative case. However, as illustrated in section 5, its feature matrix needs to allow the verb to occur in coordination constructions with dative verbs such as *helfen* 'to help'. In order to achieve this, the template has an optional slot called `:allow` in which indeterminate values can be specified:

(38)
```
(def-lex-feature-matrix findet-lex
 :feature
   (:syn-cat :syn-valence :object :case)
 :dimensions (acc)
 :allow (dat)
 :paradigm *german-case*)
```

If the `:allow` slot is filled, the template always assigns a positive value to the main dimensions that are associated with the symbols that are provided to the `:dimensions` slot. Additionally, the `:allow` slot takes a list of dimensions as well that will remain a variable instead of becoming '–'. The resulting indeterminate feature matrix is the same one as shown in the object-unit in example (25).

6.4. Agreement and Percolation

Some constructions do not subscribe their feature matrices to a grammatical paradigm by assigning positive or negative values to the cells in the matrix. Instead, they impose agreement constraints on the feature matrices of their constituent units. For example, a determiner-noun construction in German causes its determiner and head noun to agree in case, number, gender and declension class. Argument structure constructions impose agreement between the subject and main verb of a clause, and between the verb and its direct object in transitive clauses.

If all cells in the matrices need to be in agreement with each other, it suffices to use a single variable for representing the entire matrix instead of repeating it as a whole. This is the same strategy as illustrated by Steels (2011a, section 5.2) for nominal phrases. The following example uses the same template proposed by Steels for doing agreement and percolation in phrasal constructions:

(39)
```
(def-phrasal-agreement
 determiner-nominal-cxn
  (?nominal-phrase
   :syn-cat (==1 (case ?case)))
  (?determiner-unit
   :syn-cat (==1 (case ?case)))
  (?nominal-unit
   :syn-cat (==1 (case ?case))))
```

In the above, the feature matrices of the `?determiner-unit` and `?nominal-unit` both need to be bound to the variable `?case`, which means they need to unify with each other. Unification of both values involves a compatibility check and replacement of variables whenever possible. The resulting feature matrix is percolated up to the nominal phrase by repeating the same variable `?case`.

If agreement is however only required for particular dimensions of the matrix, it is necessary to define separate matrices for each unit and then use variable equalities in only those cells that need to agree with each other. This has already been schematically illustrated for German coordination constructions in Figure 6. The templates required for achieving partial agreement, however, fall beyond the scope of this paper.

7. Relation to Other Work

Feature matrices target specific linguistic phenomena that are known to be hard in unification-based grammars. Besides the aforementioned disjunctive feature representation and multiple type hierarchies, several solutions have been proposed for such phenomena that are very close in spirit to the approach in this paper. In this section, I briefly discuss these proposals and explain how they are different from feature matrices.

7.1. Ingria (1990)

Ingria (1990) was one of the first scholars to point out that complex grammatical phenomena cannot be adequately modeled as simple feature-value pairs. He first considers a solution that comes close to feature matrices and assumes "that Case [...] is not a single-valued feature, but rather an array of the different Cases of the language, each of which takes on one of the values T or NIL" (ibid. at p. 196). T stands for either underspecification (multiple Ts in one value) or a positive value; NIL stands for a negative value.

However, Ingria dismisses the solution based on examples of feature indeterminacy from Hungarian, French and German. The French and German examples concern coordination constructions such as the ones discussed in section 5. The Hungarian example involves agreement between verbs and their objects (see Beuls, 2011, for a more detailed discussion of the linguistic facts). Most Hungarian verbs are marked as definite or indefinite in agreement with their complement, which in Ingria's solution would be represented as follows for definite verb forms:

(40)
$$\left[\text{DEFINITENESS} \begin{bmatrix} \text{definite} & \text{T} \\ \text{indefinite} & \text{NIL} \end{bmatrix}\right]$$

However, some verb forms such as *akartam* 'I-wanted' are underspecified for definiteness, which, in Ingria's approach, is captured as follows:

(41) *akartam* 'I-wanted':
$$\left[\text{DEFINITENESS} \begin{bmatrix} \text{definite} & \text{T} \\ \text{indefinite} & \text{T} \end{bmatrix}\right]$$

Ingria then writes that WH-pronouns, which are also marked for definiteness (e.g. *amit* 'which.INDEF.'), need to keep one of those values *unspecified* so they can co-occur with underspecified verb forms:

(42) *amit* ‘which’:

$$\left[\begin{array}{ll}\text{DEFINITENESS} & \left[\begin{array}{ll}\text{definite} & \text{?unspecified}\\ \text{indefinite} & \text{T}\end{array}\right]\end{array}\right]$$

According to Ingria (1990, p. 197–198), unification of such a WH-pronoun with an underspecified verb form is problematic because it results in a structure in which the definiteness/indefiniteness contrast is neutralized (both values are T), even though there is no ambiguity for native speakers of Hungarian. Making a similar case for German and French coordination constructions, Ingria concludes that unification is not sufficient for handling complex agreement phenomena and proposes a different method instead based on distinctiveness checks.

So how do feature matrices relate to Ingria’s proposal? Section 5 has already shown that feature matrices falsify Ingria’s claims that unification is insufficient for dealing with phenomena such as German coordination constructions. Here, I will briefly show that they can also tackle Hungarian agreement (see Beuls, 2011, for a detailed implementation).

The main difference between Ingria (1990) and this paper is how both approaches represent underspecification. Whereas Ingria uses the value ‘T’ for both positive and underspecified values, feature matrices use ‘+’ uniquely for positive values and variables for underspecified or unspecified values (in fact, feature matrices do not distinguish underspecification from unspecification), which means that underspecified verb forms such as *akartam* ‘I-would’ (example 41) would be represented as follows:

(43) *akartam* ‘I-wanted’:

$$\left[\begin{array}{ll}\text{DEFINITENESS} & \left[\begin{array}{ll}\text{definite} & \text{?def}\\ \text{indefinite} & \text{?indef}\end{array}\right]\end{array}\right]$$

The WH-pronouns do not need an unspecified value anymore in order to unify with underspecified verbs. The pronoun *amit* ‘which.INDEF’ (example 42) would therefore look as follows:

(44) *amit* ‘which’:

$$\left[\begin{array}{ll}\text{DEFINITENESS} & \left[\begin{array}{ll}\text{definite} & -\\ \text{indefinite} & +\end{array}\right]\end{array}\right]$$

The unification of both structures now correctly predicts that Hungarian speakers would arrive at an indefinite reading of the utterance:

(45) unification of *akartam* + *amit*:

$$\begin{bmatrix} \text{DEFINITENESS} & \begin{bmatrix} \text{definite} & - \\ \text{indefinite} & + \end{bmatrix} \end{bmatrix}$$

In sum, feature matrices do not require additional mechanisms or exceptional rules for dealing with more complex phenomena such as Hungarian agreement.

7.2. Dalrymple et al. (2009)

Dalrymple et al. (2009) propose an implementation that comes closer to the feature matrices of this paper: complex features such as case are represented as an array of features that take binary values ('+' or '–'). Similar to feature matrices but opposed to Ingria (1990), a '+' is only assigned in case of positive specification. For example, the German pronoun *wer* 'who.NOM' is positively assigned nominative case:

(46) *wer* 'who':

$$\begin{bmatrix} \text{CASE} & \begin{bmatrix} \text{NOM} & + \\ \text{ACC} & - \\ \text{DAT} & - \\ \text{GEN} & - \end{bmatrix} \end{bmatrix}$$

A form like *Kinder* 'children' cannot be assigned dative case, but all three other cases are possible, which is represented through underspecification:

(47) *Kinder* 'children':

$$\begin{bmatrix} \text{CASE} & \begin{bmatrix} \text{NOM} & \\ \text{ACC} & \\ \text{DAT} & - \\ \text{GEN} & \end{bmatrix} \end{bmatrix}$$

By reserving '+' for positive values only, Dalrymple et al. (2009) can effectively overcome the problems of Ingria (1990) in the same way as explained for feature matrices in the previous subsection.

The main difference between Dalrymple et al. (2009) and this paper is that feature matrices exploit the expressive power of variables for representing underspecification (see section 3.2). Without variables, the grammarian is still forced to resort to inefficient disjunctive feature representation or type hierarchies. German examples are used again to illustrate this point.

Let's take the German definite article *der*. As shown in Table 1, the constraints we need to represent are as follows: (a) *der* can be assigned nominative case, but only if the noun is masculine singular, (b) it can be assigned dative or genitive case, but only if the noun is feminine singular, (c) it can unify with all plural nouns, but only in genitive case. Without variables, the approach of Dalrymple et al. requires a disjunctive feature representation:

(48) *der*:

$$\left\{ \begin{array}{l} \begin{bmatrix} \text{CASE} & \begin{bmatrix} \text{NOM} & + \\ \text{ACC} & - \\ \text{DAT} & - \\ \text{GEN} & - \end{bmatrix}, \begin{bmatrix} \text{NUM} & \text{SG} \end{bmatrix}, \begin{bmatrix} \text{GENDER} & \text{M} \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \text{CASE} & \begin{bmatrix} \text{NOM} & - \\ \text{ACC} & - \\ \text{DAT} & \\ \text{GEN} & \end{bmatrix}, \begin{bmatrix} \text{NUM} & \text{SG} \end{bmatrix}, \begin{bmatrix} \text{GENDER} & \text{F} \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \text{CASE} & \begin{bmatrix} \text{NOM} & - \\ \text{ACC} & - \\ \text{DAT} & - \\ \text{GEN} & + \end{bmatrix}, \begin{bmatrix} \text{NUM} & \text{PL} \end{bmatrix}, \begin{bmatrix} \text{GENDER} & \end{bmatrix} \end{bmatrix} \end{array} \right\}$$

This is opposed to the single representation using feature matrices. (see Table 3.) More importantly, feature matrices are significantly more efficient than disjunctive feature representation as they do not cause unnecessary splits in the search tree (as discussed in section 2.1).

8. Conclusions

This paper illustrated the challenges of ambiguity and feature indeterminacy for unification-based grammar formalisms. It examined two widespread, traditional techniques for dealing with those challenges: disjunctive feature representation and type hierarchies. Even though both approaches have their merits, I showed that they have several shortcomings when dealing with more complex issues such as German case agreement. More specifically, disjunctions are highly inefficient in processing. Type hierarchies are capable of resolving this problem in most cases, but when it comes to phenomena such as likeness constraints and feature indetermination in coordination, additional data structures and/or type hierarchies are needed.

As an alternative, I presented feature matrices that only use unification without resorting to the introduction of neutral features in a type hierarchy. I showed that by carefully representing the grammatical paradigm of a language (sub)system through variables, it is possible to handle even those constructions that are hard for traditional solutions.

From a theoretical point of view, feature matrices are better representatives for such linguistic phenomena than traditional disjunctions. Instead of forcing the search tree to split, they postpone commitment to a particular value until necessary. The search space will thus be more likely to reflect attested ambiguities in a language rather than ambiguities that are uniquely due to the grammarian's particular design choice. Feature matrices also make type hierarchies obsolete or at least get rid of a great deal of complexity in them: there is no need for additional data structures (such as typed lists), separate hierarchies for inherent case and indetermination, neutral features, and others. It is important to note, however, that feature matrices do not exclude for example being combined with type hierarchies. Indeed, the goal of this paper was not to argue against traditional solutions, which have proven their worth in earlier work. Feature matrices rather provide an opportunity for improving these techniques.

Even though feature matrices can be used in any unification-based formalism, I provided a specific implementation for using them in Fluid Construction Grammar. This implementation provides templates that can be generally applied to every linguistic feature that requires a feature matrix. All the examples of this paper can be verified through interactive web demonstrations at `www.fcg-net.org`.

Finally, I related the approach in this paper to similar proposals in the field. I showed that feature matrices improve on previous techniques by exploiting the expressive power of variables. By doing so, they either overcome fundamental problems of earlier approaches or they offer the grammar engineer a more elegant and efficient way of representing and processing complex grammatical phenomena.

Acknowledgements

Research reported in this paper was funded by the Sony Computer Science Laboratory Paris and the EU FP7 ALEAR project. I wish to thank Luc Steels for his feedback on this work, and the whole team working on FCG at the VUB AI-Lab (University of Brussels) and at Sony CSL Paris, particularly Sebastian Höfer for improving the implementation. I also thank the anonymous reviewers of this paper for the insightful comments and constructive feedback. All remaining errors are of course my own.

References

Beuls, Katrien (2011). Construction sets and unmarked forms: A case study for Hungarian verbal agreement. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Bresnan, Joan (Ed.) (1982). *The Mental Representation of Grammatical Relations*. Cambridge, MA: MIT Press.

Carter, David (1990). Efficient disjunctive unification for bottom-up parsing. In *Proceedings of the 13th Conference on Computational Linguistics*, 70–75. ACL.

Copestake, Ann (2002). *Implementing Typed Feature Structure Grammars*. Stanford: CSLI Publications.

Crysmann, Berthold (2005). Syncretism in German: A unified approach to underspecification, indeterminacy, and likeness of case. In Stefan Müller (Ed.), *Proceedings of the 12th International Conference on Head-Driven Phrase Structure Grammar*, 91–107. Stanford: CSLI Publications.

Dalrymple, Mary, Tracy Holloway King, Louisa Sadler (2009). Indeterminacy by underspecification. *Journal of Linguistics*, 45, 31–68.

Daniels, Michael (2001). On a type-based analysis of feature neutrality and the coordination of unlikes. In *Proceedings of the 8th International Conference on HPSG*, 137–147. Stanford: CSLI.

Flickinger, Daniel P. (2000). On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1), 15–28.

Gazdar, Gerald, Ewan Klein, Geoffry Pullum, Ivan Sag (1985). *Generalized Phrase Structure Grammar*. Oxford: Basil Blackwell.

Heinz, Wolfgang, Johannes Matiasek (1994). Argument structure and case assignment in german. In John Nerbonne, Klaus Netter, Carl Pollard (Eds.), *German in Head-Driven Phrase Structure Grammar*, *CSLI Lecture Notes*, vol. 46, 199–236. Stanford: CSLI Publications.

Ingria, Robert (1990). The limits of unification. In *Proceedings of the 28th Annual Meeting of the ACL*, 194–204.

Karttunen, Lauri (1984). Features and values. In *Proceedings of the 10th International Conference on Computational Linguistics*. Stanford.

Kay, Martin (1985). Parsing in Functional Unification Grammar. In David Dowty, Lauri Karttunen, Arnold Zwicky (Eds.), *Natural Language Parsing*. Cambridge: Cambridge University Press.

Levine, Robert, Thomas Hukari, Michael Calcagno (2001). Parasitic gaps in english: Some overlooked cases and their theoretical consequences. In Peter W. Culicover, Paul M. Postal (Eds.), *Parasitic Gaps*. Cambridge MA: MIT Press.

Micelli, Vanessa (2012). Field topology and information structure - a case study for German constituent order. In Luc Steels (Ed.), *Computational Issues in Fluid Construction Grammar*. Berlin: Springer.

Müller, Stefan (1999). An HPSG-analysis for free relative clauses in german. *Grammars*, 2(1), 53–105.

Müller, Stefan (2001). Case in German – towards and HPSG analysis. In Tibor Kiss, Detmar Meurers (Eds.), *Constraint-Based Approaches to Germanic Syntax*. Stanford: CSLI.

Pullum, Geoffrey, Arnold Zwicky (1986). Phonological resolution of syntactic feature conflict. *Language*, 62(4), 751–773.

Ramsay, Allan (1990). Disjunction without tears. *Computational Linguistics*, 16(3), 171–174.

Sag, Ivan A. (2003). Coordination and underspecification. In Jongbok Kom, Stephen Wechsler (Eds.), *Proceedings of the Ninth International Conference on HPSG*. Stanford: CSLI.

Spranger, Michael, Martin Loetzsch (2011). Syntactic indeterminacy and semantic ambiguity: A case study for German spatial phrases. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Steels, Luc (2011a). A design pattern for phrasal constructions. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Steels, Luc (2011b). A first encounter with Fluid Construction Grammar. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Steels, Luc, Pieter Wellens (2006). How grammar emerges to dampen combinatorial search in parsing. In P. Vogt, Y. Sugita, E. Tuci, C. Nehaniv (Eds.), *Symbol Grounding and Beyond. Proceedings of the Third EELC*, LNAI 4211, 76–88. Berlin: Springer-Verlag.

van Trijp, Remi (2011). A design pattern for argument structure constructions. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.