# Notice

*This paper is the author's draft and has now been published officially as:*

Höfer, Sebastian (2012). Complex Declension Systems and Morphology in Fluid Construction Grammar: A Case Study of Polish. In Luc Steels (Ed.), *Computational Issues in Fluid Construction Grammar*, 143–177. Berlin: Springer.

*BibTeX:*

```
@incollection{hoefer2012complex,
   Author = {Hoefer, Sebastian},
   Title = {Complex Declension Systems and Morphology in Fluid
            Construction Grammar: A Case Study of Polish},
   Pages = {143--177},
   Editor = {Steels, Luc},
   Booktitle = {Computational Issues in {Fluid Construction Grammar}},
   Publisher = {Springer},
   Series = {Lecture Notes in Computer Science},
   Volume = {7249},
   Address = {Berlin},
   Year = {2012}}
```

# Complex Declension Systems and Morphology in Fluid Construction Grammar: A Case Study of Polish

Sebastian Höfer

Robotics and Biology Laboratory, Technische Universität Berlin, Germany

**Abstract.** Different languages employ different strategies for grammatical agreement. Slavic languages such as Polish realize agreement with rich declension systems. The Polish declension system features seven cases, two number categories and is subdivided further with respect to gender and animacy. In order to differentiate among these different grammatical categories Polish exhibits a complex, syncretistic and highly irregular morphology. But not only the morphology is complex, the grammatical rules that govern agreement are, too. For example, the appropriate case of a noun in a verbal phrase does not only depend on the verb itself but also on whether the verb is in the scope of a negation or not.
In this paper we give an implementation of the Polish declension system in Fluid Construction Grammar. In order to account for the complexity of the Polish declension system we develop a unification-based formalism, called *nested feature matrices*. To demonstrate the power of the proposed formalism we investigate its appropriateness for solving the following linguistic problems: a) selecting appropriate morphological markers with respect to the noun's gender and stem for expressing case and number, b) establishing phrasal agreement between nouns and other parts of speech such as verbs, and finally c) dealing with long-distance dependencies in phrasal agreement. We show that our formalism succeeds in solving these problems and that the presented implementation is fully operational for correctly parsing and producing simple Polish transitive sentences.
[1, 2]. This grammar formalism was developed in order to study the acquisition and evolution of language in computer simulations [3]. In this paper, we show that FCG can be used to model the complex and highly irregular Polish noun declension scheme. In order to model morphological and phrasal agreement, our approach makes use of a unification-based formalism called feature matrices. These are explained here in detail and are extended in a canonical way to nested feature matrices, which overcome certain drawbacks of simple feature matrices.

## 1 Introduction

Developing computational models for Slavic languages is very beneficial for understanding language in general, as Slavic languages exhibit many complex grammatical and morphological structures that English and other Western European

languages lack. A particularly intriguing part of Slavic languages is their declension system. The Polish declension system, for instance, features seven cases, two number categories and is subdivided further with respect to gender and animacy. In order to differentiate among these different grammatical categories Polish exhibits a complex, syncretistic and highly irregular morphology. But not only the morphology is complex, the grammatical rules that govern agreement are, too. For example, the appropriate case of a noun in a verbal phrase does not only depend on the verb itself but also on whether the verb is in the scope of a negation or not.

Due to their inherent linguistic complexity, there has been growing interest in formalizing Slavic languages in computational grammar theories in recent years [4, 5]. One of the currently most prominent grammar formalisms is Head-Driven Phrase Structure grammar (HPSG) [6] and therefore most of the Slavic community focussed on the implementation of Slavic languages theories in this formalism.

In this paper, the implementation of different aspects of the Polish noun declension system in Fluid Construction Grammar (FCG) is presented. The main purpose is to show that the FCG formalism provides a uniform way of dealing with the Polish declension system at the morphological, syntactical and semantical level. In order to account for the complexity of the Polish declension system we develop a unification-based formalism, called *nested feature matrices*. To demonstrate the power of the proposed formalism we investigate its appropriateness for solving the following linguistic problems: a) selecting appropriate morphological markers with respect to the noun's gender and stem for expressing case and number, b) establishing phrasal agreement between nouns and other parts of speech such as verbs, and finally c) dealing with long-distance dependencies in phrasal agreement in terms of a weak form of the so-called long distance genitive of negation. We show that our formalism succeeds in solving these problems and that the presented implementation is fully operational for correctly parsing and producing simple Polish transitive sentences.

The work presented in this paper builds upon many studies concerning the operationalization of lexical, phrasal and morphological constructions in FCG. It follows the FCG design pattern approach presented in this volume [7], and shares the implementation of morphological constructions with operationalizations of Hungarian verbs [8], Spanish modals [9] and Russian verbal aspect [10]. Furthermore, we develop an extension of the feature matrix formalism which has been originally introduced in [11] for dealing with syncretisms in the German declension system.

As the Slavic community has conducted a lot of work on the implementation of Slavic linguistic phenomena in HPSG, the reader might also refer to another chapter in this volume presenting FCGLight [12]. This formalism makes a link between HPSG and FCG by implementing a core subset of the latter in LIGHT, a system previously used for the implementation of large-scale HPSG grammars [13]. Another related study on a Slavic language example in FCG which deals with verbal aspect in Romanian can be found in [14].

The remainder of this paper is structured as follows: first, the linguistic problems addressed in this paper are explained. Next, feature matrices and nested feature matrices are introduced, which constitute the core formalism used to model the declension system and establish agreement on the morphological and syntactic level. Finally, the operationalization of the Polish case study in FCG is presented, and the results are summarized and evaluated.

## 2 Linguistic Insights

This case study presents a formalization of the Polish noun declension system, the so-called genitive of negation and the long distance genitive of negation. The main part of the implementation deals with morphological aspects of the Polish noun declension system, and, in particular, how this complex system can be represented in FCG in a uniform manner. The long distance genitive of negation phenomenon is considered in order to illustrate that FCG is also suitable for the operationalization of long distance dependencies. In order to grasp the implementational details of the formalism presented later, some linguistic background on these phenomena is required which is provided in the following section.

### 2.1 Polish Noun Declension System

Nominal inflections in Polish conflate two grammatical categories: case and number. There are seven cases (nominate, genitive, dative, accusative, instrumental, locative and vocative) and two numbers (singular and plural). Additionally, Polish nouns are traditionally divided into three inflectional paradigms or *declension schemes* in terms of the three genders (masculine, feminine, neuter). However, the paradigms are not entirely consistent; for almost every case and number there exist several endings, and the appropriate ending depends mainly on the morphological and syntactic properties of the noun. Therefore the number of distinct paradigms is very large [15]. Things get even more complicated, since some nouns can have two different endings in the genitive case, depending on the meaning that a speaker wants to express.

Let us look at an example in order to illustrate the complexity of the Polish declension system. Consider the following examples, which deal with the two masculine nouns *przypadek* (case) and *człowiek* (man):

(1) *W* *żadnym* ***przypadku*** *nie* *znajdziemy* *całej* *prawdy*
In no.LOC case.LOC not (we) find whole.GEN truth.GEN
*o* ***człowieku***.
about man.LOC
'In no case do we find out the whole truth about mankind.'

This example shows that *-u* is a marker for the locative case for masculine nouns, and indeed it is so quite consistently. The next examples consider the *-a* ending:

(2) *Nie ma ani jednego* ***człowieka.***
Not is even one.GEN man.GEN
'There is not even one man.'

(3) *Widzę jednego* ***człowieka.***
(I) see one.ACC man.ACC.
'I see one man.'

(4) *Przyimek 'zamiast' wymaga użycia drugiego*
Preposition.NOM 'zamiast' requires usage.GEN second.GEN
***przypadka.***
case.GEN
'The preposition 'zamiast' requires usage of the second case.'

(5) *Pierwszy raz widzę taki* ***przypadek.***
first time (I) see such.ACC case.ACC
'It is the first time that I see such a case.'

The first thing to be noticed is that the ending *-a* occurs with both nouns, but not always in the same cases. The examples 2 and 4 suggest that *-a* serves as a marker for genitive. However, things are more complicated: as seen in example 3, the accusative of *człowiek* corresponds to its genitive, that is, it takes the *-a* too. On the other hand, the accusative of *przypadek* corresponds to its nominative and is therefore unmarked. Why is this so? As a rule of thumb, nouns that denote virile (masculine-human) individuals agree in the genitive and the accusative case, while nouns denoting non-animate objects agree in nominative and accusative. Therefore, in many Polish textbooks the masculine gender is subdivided into three: virile (animate and personal), animate (and not personal) and impersonal. In current Polish language the amount of nouns following the virile scheme is steadily expanding [15]; particularly neologisms like *email* follow the virile declension scheme and therefore exhibit the *-a* marker in accusative, although they denote inanimate objects.

Yet, the whole issue of the declension scheme becomes even more puzzling, as the following example shows:

(6) *Nie było takiego* ***przypadku*** *choroby.*
Not was such.GEN case.GEN illness.GEN.
'There was no such case of illness.'

The noun *przypadek* is an example from a small group of nouns which can take either *-a* or *-u* in the genitive case. Whether one or the other ending is preferred sometimes depends on idiomatic use. In this case it depends on the intended meaning of the word: if the grammatical case is meant, as in example 4, the noun takes the *-a* marking. If a certain circumstance or fact is to be expressed, the *-u* marking is used. See, for example, [16] for an extensive discussion of the distribution of the two endings in genitive masculine.

Table 1 exemplifies in which cases the ending *-a* can actually occur. It shows that the marker is completely independent from the grammatical categories case, number and gender.

| *-a* | **SG** | | | **PL** | | |
|---|---|---|---|---|---|---|
| **Case** | **SG-M** | **SG-F** | **SG-N** | **PL-M** | **PL-F** | **PL-N** |
| *NOM* | *sędzia* (judge) | *dziewczyna* (girl) | – | *księża* (dukes) | – | *pola* (fields) |
| *GEN* | *człowieka* (man) | – | *pola* (field) | – | – | – |
| *DAT* | – | – | – | – | – | – |
| *ACC* | *człowieka* | – | – | *cuda* (miracles) | – | *pola* |
| *INS* | – | – | – | – | – | – |
| *LOC* | – | – | – | – | – | – |
| *VOC* | – | – | – | – | – | *pola* |

**Table 1.** The ending *-a* occurs across different cases, genders and numbers. An example of a noun form is given for each of the possible entries occurrences of the ending. The nominative singular forms of the inflected nouns are *książe* (duke), *pole* (field), *człowiek* (man) and *cud* (miracle).

**Stem Palatalization** As in many other Slavic languages, during the evolution of Polish several sound changes subsumed under the term *palatalization* occurred. In this process mid, close front vowels (e.g. /i/, /e/) and the semi-vowel /j/ shift nearby phonemes, usually preceding consonants, towards the palatal articulatory position. There exist several types of palatalization, such as the so-called *iotation*, as in the occurrence of the *i* in *nie* (no / not) results in changing the sound of /n/ to /ɲ/. In Polish, the /ɲ/ phoneme is represented by *ni* in the beginning and by *ń* in the end of a syllable. Consider, for example, *dzień* (day) and *nie*, which both contain this very same phoneme.

The Polish declension system includes two palatalizing endings denoted by *-'e* and *-'i*. The way the stem of a noun is changed depends on the stem or the stem consonant of a noun, respectively. Hence, consider for example the changes that occur in feminine nouns that exhibit the *-'e* in the dative singular case: *ryba* becomes *rybie* (fish), *łąka* becomes *łące* (meadow), *skóra* becomes *skórze*, etc. There do exist reliable rules as to how palatalization affects nearby phonemes. For a more complete analysis see, for example, [17].

Note that the non-palatalizing counterparts *-e* and *-i* also appear in the declension scheme, for example, the dative singular of the noun *szansa* (chance) takes the palatalized ending and becomes *szansie*, but the noun's nominative plural is *szanse*.

In any case, this brief analysis shows that beside the support for parsing and production of nouns and their endings, an operationalization of the Polish declension scheme must also account for changes which affect the stem of a noun.

### 2.2 Genitive of Negation

Another interesting grammatical phenomenon in Slavic languages, including Polish, is the so-called *genitive of negation (GoN)*. The phenomenon can be explained as follows: in the presence of verbal negation, the genitive case is assigned to the argument of the verb if the verb requires the argument to take the accusative in the absence of negation. The following example demonstrates that the case of the accusative object needs to be changed due to the appearance of the negative marker *nie*:

(7) *Michał* *widzi* ***Marię.***
Michael.NOM sees Maria.ACC
'Michael sees Maria.'

(8) *Michał* *nie* *widzi* ***Marii.***
Michael.NOM not sees Maria.GEN
'Michael does not see Maria.'

Note that the GoN only applies to accusative objects:

(9) *Michał* *macha* ***patykiem.***
Michael.NOM waves stick.INS.
'Michael waves the stick.'

(10) *Michał* *nie* *macha* ***patykiem.***
Michael.NOM not sees stick.INS
'Michael does not wave the stick.'

Moreover, the GoN does not only affect finite verb forms but also non-finite verb forms such as infinitivals and participles.

Whereas in languages such as Russian, the application of GoN is not obligatory but often depends on pragmatic, semantic or idiosyncratic factors, GoN in Polish is fully grammaticalized, that is, it is triggered by the morphosyntactic structure of the negative marker *nie*.[1]

### 2.3 Long Distance GoN

An interesting property of the GoN in Polish is that it appears even in cases where the *nie* does not negate the verb directly, but only a verb higher in the hierarchical structure of the sentence [19]. This is illustrated in the following example, where the negation applies to the auxiliary verb but still causes the object of the infinitival to take genitive case:

[1] However, as already noted in [18], there exist some exceptional cases in Polish where the GoN is indeed optional, for example [19]:

(11) *Marię / Marii* *nie boli* *głowa.*
Maria.ACC / Maria.GEN not aches head.NOM
'Maria does not have a headache.'

(12) *Michał* *nie* *chce* *widzieć* ***Marii.***
Michael.NOM not want see Maria.GEN
'Michael does not want to see Maria.'

This phenomenon is discussed in more detail in [19] and additional cases are presented where the long distance GoN is not obligatory or is singled out by Polish native speakers. In the following analysis, a simple example consisting of an auxiliary and an infinitival as shown above is considered. In spite of its simplicity, it reflects well how long distance relationships can be handled in FCG.

## 3 Feature Matrices

The previous section demonstrated the need for an efficient formalism that is able to account for the complexity of the presented declension system. The approach in this paper is based on *feature matrices* which constitute a solely unification based method for handling syncretisms in terms of case, gender and number distinctions [11]. Common accounts for the issue of case syncretism involve *disjunctive feature representations* which represent the multifunctionality by disjunctions, i.e. alternatives. Let us contrast the two approaches with an example and consider the representations for the proper name *Maria* and the *-a* ending. To be more precise, not the feature matrix of *Maria*, but only of its stem *Mari-* is considered, since the former already is a combination of a noun stem with an ending. The noun stem can be represented by the following concise disjunctive feature representation:

*Mari-*:

$$\begin{bmatrix} \text{GENDER} & f \\ \text{NUM} & sg \\ \text{CASE} & nom \vee gen \vee dat \vee acc \vee ins \vee loc \vee voc \end{bmatrix}$$

Since *Mari-* is considered a noun stem, all cases are allowed and have to be specified either by being left unmarked or by acquiring a suffix. Because *Maria* is a female proper name, this example assumes that it only occurs in the singular case. The representation for the ending *-a* looks as follows:

*-a*:

$$\begin{bmatrix} \text{GENDER} & m \\ \text{NUM} & sg \\ \text{CASE} & nom \vee gen \vee acc \end{bmatrix} \vee \begin{bmatrix} \text{GENDER} & m \\ \text{NUM} & pl \\ \text{CASE} & nom \vee acc \end{bmatrix} \vee$$

$$\begin{bmatrix} \text{GENDER} & f \\ \text{NUM} & pl \\ \text{CASE} & nom \end{bmatrix} \vee \begin{bmatrix} \text{GENDER} & n \\ \text{NUM} & sg \\ \text{CASE} & gen \end{bmatrix} \vee \begin{bmatrix} \text{GENDER} & n \\ \text{NUM} & pl \\ \text{CASE} & nom \vee acc \end{bmatrix}$$

The respective feature matrix representation for *Mari-* is shown in Table 2.

| *Mari-* | | **SG** | | | **PL** | | |
|---|---|---|---|---|---|---|---|
| **Case** | **C** | **SG-M** | **SG-F** | **SG-N** | **PL-M** | **PL-F** | **PL-N** |
| NOM | *?n* | *?n-sg-m* | *?n-sg-f* | – | *?n-pl-m* | – | *?n-pl-n* |
| GEN | *?g* | *?g-sg-m* | – | *?g-s-n* | – | – | – |
| DAT | – | – | – | – | – | – | – |
| ACC | *?a* | *?a-sg-m* | – | – | *?a-pl-m* | – | *?a-pl-n* |
| INS | – | – | – | – | – | – | – |
| LOC | – | – | – | – | – | – | – |
| VOC | *?v* | *?v-sg-m* | – | – | – | – | *?v-pl-n* |

**Table 2.**

Table 3 shows the feature matrix for the ending *-a* which can directly be read from Table 1 on page 6.

| *-a* | | **SG** | | | **PL** | | |
|---|---|---|---|---|---|---|---|
| **Case** | **C** | **SG-M** | **SG-F** | **SG-N** | **PL-M** | **PL-F** | **PL-N** |
| NOM | *?n* | *?n-sg-m* | *?n-sg-f* | – | *?n-pl-m* | – | *?n-pl-n* |
| GEN | *?g* | *?g-sg-m* | – | *?g-s-n* | – | – | – |
| DAT | – | – | – | – | – | – | – |
| ACC | *?a* | *?a-sg-m* | – | – | *?a-pl-m* | – | *?a-pl-n* |
| INS | – | – | – | – | – | – | – |
| LOC | – | – | – | – | – | – | – |
| VOC | *?v* | *?v-sg-m* | – | – | – | – | *?v-pl-n* |

**Table 3.**

In the given feature matrix representations, strings preceded by a quotation mark denote *variables* which can be bound during the unification process to either '–', '+', or other variables. As opposed to variables, strings without quotation marks are called *symbols*, such as 'NOM', '–' and '+'. Additionally, the matrix cells contained in the columns beginning with the *SG-M* column are called *feature cells*. The symbol '–' occurring in a feature cell means that the particular case-gender-number combination is not possible for the linguistic item. A variable displays the possibility of this combination, and the '+' symbol makes a final commitment, determining the grammatical categories of the item. The upcoming example shows how a well-formed feature matrix should look and how feature matrix unification works.

Whereas disjunctive features prove elegant in easily separable cases without intermingling categories like for the form *Mari-*, they obviously do not form a very compact representation for irregular systems as the Polish case system, as can be seen for the *-a* ending. Feature matrices, on the other hand, may be fairly

sparse for very specific linguistic items. However, disjunctions were proven to be computationally expensive [20]. Moreover, unification of disjunctive features is, in general, NP-complete [21]. In contrast, feature matrix unification is a rather simple task in terms of computational effort. Unification based processing of disjunctive features cannot be covered in detail at this point, since this paper concentrates solely on feature matrices. For a more detailed comparison, see [11].

Let us take a look at the unification result of the feature matrices for *Mari-* and *-a*. What exactly happens during unification? The formal details of unifi-

| *Mari-a* | | **SG** | | | **PL** | | |
|---|---|---|---|---|---|---|---|
| **Case** | **C** | **SG-M** | **SG-F** | **SG-N** | **PL-M** | **PL-F** | **PL-N** |
| NOM | *?n-sg-f* | – | *?n-sg-f* | – | – | – | – |
| GEN | – | – | – | – | – | – | – |
| DAT | – | – | – | – | – | – | – |
| ACC | – | – | – | – | – | – | – |
| INS | – | – | – | – | – | – | – |
| LOC | – | – | – | – | – | – | – |
| VOC | – | – | – | – | – | – | – |

**Table 4.**

cation and merging in FCG are given in [22], however, an intuitive notion of unification of feature matrices is given in the following.

All cells of both matrices are compared pairwise, that is, the cell in row $i$, column $j$ from the first matrix is compared to the cell in row $i$, column $j$ from the second matrix. If both cells contain symbols (in this case a string like 'NOM', '–' or '+'), they must be identical; if one cell contains a variable and the other a symbol, the variable is bound to this symbol – if the variable is not bound to another symbol yet. Similarly, if both cells contain variables, the first variable is bound to the second one. Thus, for example, the variable *?n-sg-m* from the *-a*-matrix is bound to '–', since this is the value of the corresponding cell in the *Mari-* matrix. Obviously, as shown in Table 4 the right solution is obtained, because all possibilities except nominative singular feminine get sorted out this way.

At this point, the role of the matrix's second column, *C*, has not yet been explained. The *C* actually stands for *case* and will be called the *case column*. In order to understand what that case column does, let us take a look at the resulting *Mari-a* matrix in Table 4 again. The feature cell corresponding to nominative singular feminine contains the same variable *?n-sg-f* as the cell in the *C* column in the nominative row. That means that if *?n-sg-f* is bound to '+' or '–', both cells are affected. It was mentioned before that a '+' would signal full commitment; obviously, the result is unique, but why does no '+' occur in the resulting matrix then? The reason is that neither of the two linguistic items, neither the noun stem nor the suffix, can make a definite commitment for a case on its own. Assuming that the *-a* ending would only occur in nominative, there

would be a '+' in the nominative case column of the *-a* feature matrix, and this '+' would also make its way into the result matrix through unification.

Whether to put a '+' in a feature cell, too, depends on the number of alternatives in this row: if the item can mark singular and plural or different genders, there would be variables like before. Otherwise, if a specific case-gender commitment could be made, there would be a '+' in the corresponding column as well.

Although in the *Mari-a* example no final commitment in terms of '+'s could be made, it is not harmful in practice when constructions that commit to case are involved. This is illustrated by the following sentence:

(13) *Maria* *śpi.*
Maria.NOM sleeps.
'Maria sleeps / is sleeping.'

In parsing or producing this sentence the construction for the verb *śpi* would contain a feature matrix for phrasal agreement, which would specify the case of the subject (also usually including feature matrices for the cases of potential objects, shown later). The verb's feature matrix related to the subject or agent in this phrase is shown in Table 5.

| $śpi_{Subject}$ | | **SG** | | | **PL** | | |
|---|---|---|---|---|---|---|---|
| **Case** | **C** | **SG-M** | **SG-F** | **SG-N** | **PL-M** | **PL-F** | **PL-N** |
| NOM | + | *?n-sg-m* | *?n-sg-f* | *?n-sg-n* | *?n-pl-m* | *?n-pl-f* | *?n-pl-n* |
| GEN | – | – | – | – | – | – | – |
| DAT | – | – | – | – | – | – | – |
| ACC | – | – | – | – | – | – | – |
| INS | – | – | – | – | – | – | – |
| LOC | – | – | – | – | – | – | – |
| VOC | – | – | – | – | – | – | – |

**Table 5.**

Clearly evident, when the $śpi_{Subject}$ feature matrix and the *Mari-a* feature matrix are unified, the resulting matrix contains a '+' in the nominative case column and another '+' in the feature cell corresponding to nominative singular feminine. This example illustrates the purpose of the case column: the hypothesis is that information about the case comes from the wider context, such as a verb requiring its arguments to taking specific cases, while information about number and gender can be inferred from the noun or at least from a nominal phrase. The latter is the case for languages which mainly mark case by articles like German.

### 3.1 Limitations of Feature Matrices

The last section explained the basic idea and application mechanism of feature matrices. This section will show that the previously introduced feature matrix

formalism is not flawless. However, the issues can be remedied by a canonical extension called *nested feature matrices* in the forthcoming section.

Revisiting the previous example, the *-a* is combined with a masculine noun, choosing *człowiek* (man), which is also a noun stem since the nominative case is unmarked here. For the sake of completeness, provided are both the feature matrix (Table 6) and the disjunctive feature representation for this noun:

*człowiek*:

$$\begin{bmatrix} \text{GENDER} & m \\ \text{NUM} & sg \vee pl \\ \text{CASE} & nom \vee gen \vee dat \vee acc \vee ins \vee loc \end{bmatrix}$$

| *człowiek* | | **SG** | | | **PL** | | |
|---|---|---|---|---|---|---|---|
| **Case** | **C** | **SG-M** | **SG-F** | **SG-N** | **PL-M** | **PL-F** | **PL-N** |
| NOM | *?n* | *?n-sg-m* | – | – | *?n-pl-m* | – | – |
| GEN | *?g* | *?g-sg-m* | – | – | *?g-pl-m* | – | – |
| DAT | *?d* | *?d-sg-m* | – | – | *?d-pl-m* | – | – |
| ACC | *?a* | *?a-sg-m* | – | – | *?a-pl-m* | – | – |
| INS | *?i* | *?i-sg-m* | – | – | *?i-pl-m* | – | – |
| LOC | *?l* | *?l-sg-m* | – | – | *?l-pl-m* | – | – |
| VOC | *?v* | *?v-sg-m* | – | – | *?v-pl-m* | – | – |

**Table 6.**

As before, the feature matrices for the noun stem and the ending *-a* whose feature matrix was shown in Table 3 are unified. The result is shown in Table 7.

| *człowiek-a* | | **SG** | | | **PL** | | |
|---|---|---|---|---|---|---|---|
| **Case** | **C** | **SG-M** | **SG-F** | **SG-N** | **PL-M** | **PL-F** | **PL-N** |
| NOM | *?n* | *?n-sg-m* | – | – | *?n-pl-m* | – | – |
| GEN | *?g* | *?g-sg-m* | – | – | – | – | – |
| DAT | – | – | – | – | – | – | – |
| ACC | *?a* | *?a-sg-m* | – | – | – | *?a-pl-m* | – |
| INS | – | – | – | – | – | – | – |
| LOC | – | – | – | – | – | – | – |
| VOC | *?v* | *?v-sg-m* | – | – | *?v-pl-m* | – | – |

**Table 7.**

Notice that there are many variables, thus many possibilities are still open. Next, *człowiek-a* is embedded into a full sentence:

(14) *Nie widzę człowieka.*
Not (I) see man.GEN
'I do not see the man.'

As known from Section 2.2, a negated verb calls for the genitive of negation, therefore *człowiek-a* definitely takes the genitive case. Suppose that *nie widzę* has a feature matrix for its direct object as in Table 5 (the subject matrix for *śpi*), but it contains a '+' and variables in the genitive row instead of the nominative row. If now the matrix for *człowiek-a* from Table 7 and the object feature matrix of *nie widzę* are unified, the resulting feature matrix looks as shown in Table 8.

| *(nie widzę) człowiek-a* | | **SG** | | | **PL** | | |
|---|---|---|---|---|---|---|---|
| **Case** | **C** | **SG-M** | **SG-F** | **SG-N** | **PL-M** | **PL-F** | **PL-N** |
| NOM | – | – | – | – | – | – | – |
| GEN | + | *?g-sg-m* | – | – | – | – | – |
| DAT | – | – | – | – | – | – | – |
| ACC | – | – | – | – | – | – | – |
| INS | – | – | – | – | – | – | – |
| LOC | – | – | – | – | – | – | – |
| VOC | – | – | – | – | – | – | – |

**Table 8.**

Something strange has occurred, namely that the case column and the feature cell for the genitive singular masculine do not match. Why did this happen? By checking all the feature matrices involved so far, it can be seen that none of these matrices contain the same variable in the genitive case column and the regarded feature cell. Therefore, these two cells cannot be related to each other in the final result. In fact, it is not possible for any matrices to make the genitive case column cell equal to a genitive feature cell since each matrix allows more than one possibility in the genitive case – although after the application of *all* the constructions, only one possibility remains.

The reader may wonder if this behavior poses an actual problem, which, unfortunately, indeed it does for production. In production processing goes the other way around, that is, first the phrase structure with its agreement and dependencies is processed, and the selection of the appropriate ending is made in the very end. Thus, before any ending is added to *człowiek*, the feature matrix looks like in Table 8. Now imagine that there are not one possible ending for the genitive but many different ones. To take a concrete example, let us look only at the genitive row of the feature matrix of the ending *-ów* which marks genitive and accusative plural masculine and neuter:

Now comes the problem: Although the ending *-ów* is not allowed for the masculine singular, its feature matrix unifies with the one from Table 8. More precisely, the variable *?g* will be bound to '+' and all others will be bound to

| Case | C | SG-M | SG-F | SG-N | PL-M | PL-F | PL-N |
|---|---|---|---|---|---|---|---|
| GEN | *?g* | – | – | – | *?g-pl-m* | – | *?g-pl-n* |

| | SG | | | | PL | | | |
|---|---|---|---|---|---|---|---|---|
| **Case** | **S** | **S-M** | **S-F** | **S-N** | **PL** | **PL-M** | **PL-F** | **PL-N** |
| *?n* | *?n-s* | *?n-s-m* | *?n-s-f* | *?n-s-n* | *?n-pl* | *?n-pl-m* | *?n-pl-f* | *?n-pl-n* |
| *?g* | *?g-s* | *?g-s-m* | *?g-s-f* | *?g-s-n* | *?g-pl* | *?g-pl-m* | *?g-pl-f* | *?g-pl-n* |
| *?d* | *?a-s* | *?a-s-m* | *?a-s-f* | *?a-s-n* | *?a-pl* | *?a-pl-m* | *?a-pl-f* | *?a-pl-n* |
| *?a* | *?d-s* | *?d-s-m* | *?d-s-f* | *?d-s-n* | *?d-pl* | *?d-pl-m* | *?d-pl-f* | *?d-pl-n* |
| *?i* | *?i-s* | *?i-s-m* | *?i-s-f* | *?i-s-n* | *?i-pl* | *?i-pl-m* | *?i-pl-f* | *?i-pl-n* |
| *?l* | *?l-s* | *?l-s-m* | *?l-s-f* | *?l-s-n* | *?l-pl* | *?l-pl-m* | *?l-pl-f* | *?l-pl-n* |
| *?v* | *?v-s* | *?v-s-m* | *?v-s-f* | *?v-s-n* | *?v-pl* | *?v-pl-m* | *?v-pl-f* | *?v-pl-n* |

**Table 9.** A nested feature matrix for Polish.

'–'. The meaning of the resulting feature matrix can be read as "the linguistic item takes the genitive case but has no gender and number" – which obviously does not make any sense for nouns in Polish[2].

### 3.2 Nested Feature Matrices

Before a way to solve the issue raised in the previous paragraphs is presented, the reason for the problem should be formulated in a more abstract way. As mentioned before, Polish knows three grammatical categories for nouns, case, gender and number. Thus, it can be stated that the grammatical category of a Polish noun is threefold or three-dimensional. A feature matrix, on the other hand, is in this sense only *two*-dimensional: the rows encode different cases, but the columns have to encode gender *and* number. Let us look at the feature matrix for *człowiek* in Table 6 again: the noun *człowiek* fully determines the gender, but this fact cannot be explicitly expressed in this formalism. The intuition is that a *gender column* or a *number column* are needed, which can be related to the feature cells.

In fact, this is the basic idea of *nested feature matrices*. Table 9 introduces two extra columns for number – of course gender could also be used, but then one more column would have to be added, since there are three genders but only two numbers. Now the trick is the following: if a construction containing a feature matrix can make a commitment to gender, it makes the gender column and the feature cell equal. Therefore, the nested version for *człowiek* appears as shown in Table 10. Of course, all of the other feature matrices have to be transformed to their nested versions as well.

Notice that also commitment to number can easily be modeled without the need for nesting number separately. The only requirement is to make the case and number columns equal.

[2] In Polish, even indefinite pronouns like *nic* (nothing) have to be declined.

| *człowiek* | **SG** | | | | **PL** | | | |
|---|---|---|---|---|---|---|---|---|
| **C** | **S** | **S-M** | **S-F** | **S-N** | **PL** | **PL-M** | **PL-F** | **PL-N** |
| *?n* | *?n-s-m* | *?n-s-m* | – | – | *?n-pl-m* | *?n-pl-m* | – | – |
| *?g* | *?g-s-m* | *?g-s-m* | – | – | *?g-pl-m* | *?g-pl-m* | – | – |
| *?d* | *?a-s-m* | *?a-s-m* | – | – | *?a-pl-m* | *?a-pl-m* | – | – |
| *?a* | *?d-s-m* | *?d-s-m* | – | – | *?d-pl-m* | *?d-pl-m* | – | – |
| *?i* | *?i-s-m* | *?i-s-m* | – | – | *?i-pl-m* | *?i-pl-m* | – | – |
| *?l* | *?l-s-m* | *?l-s-m* | – | – | *?l-pl-m* | *?l-pl-m* | – | – |
| *?v* | *?v-s-m* | *?v-s-m* | – | – | *?v-pl-m* | *?v-pl-m* | – | – |

**Table 10.**

In summary, in order for the resulting nested feature matrix to be well-formed and to uniquely determine case, feature and number of a syntactic form, for each of the syntactic categories, there must be at least one nested feature matrix that determines its value.

To illustrate the whole application chain, let us look at the genitive row of the nested version of the *-a* ending (Table 11).

| | **SG** | | | | **PL** | | | |
|---|---|---|---|---|---|---|---|---|
| **C** | **S** | **S-M** | **S-F** | **S-N** | **PL** | **PL-M** | **PL-F** | **PL-N** |
| *?g-sg* | *?g-sg* | *?g-sg-m* | – | *?g-sg-n* | – | – | – | – |

**Table 11.**

The commitment to number is expressed by using the same variable *?g-sg* in the case and the singular column. Finally, the genitive row of the result looks as follows (all the cells in the other rows are '-'):

| | **SG** | | | | **PL** | | | |
|---|---|---|---|---|---|---|---|---|
| **C** | **S** | **S-M** | **S-F** | **S-N** | **PL** | **PL-M** | **PL-F** | **PL-N** |
| + | + | + | – | – | – | – | – | – |

**Table 12.**

The first '+' means that the case is genitive and is introduced by the *nie widzę* object feature matrix. The '+' in the second column appears due to the fact that the case column is made equal to the singular number column by the *-a* ending matrix (Table 11). Eventually, the last '+' is in place because the *człowiek* matrix (Table 10) links the number columns to the masculine feature cells.

### 3.3 Modeling the Case System

The previous demonstrated how nested feature matrices can be used for dealing with commitment and making it explicit. Before the presentation of the actual FCG implementation for Polish, there is one last issue to solve which is related to ambiguity. Consider the following case which is the positive version of example 14:

(15) *Widzę człowieka.*
(I) see man.ACC
'I see the man.'

It was previously mentioned that virile nouns take the same form in the genitive and accusative. Let us take a look at the accusative row of the feature matrix for the ending *-a*, shown in Table 13, this time in nested form. Notice that no number commitment can be made, since both the singular and plural are allowed. However, the noun *człowiek* cannot take the *-a* ending in plural: In fact, the accusative plural entry refers to another class of masculine nouns such as *cud* (miracle). So the question is, how can this suffix be prevented from applying to the wrong noun?

| *-a* | **SG** | | | | **PL** | | | |
|---|---|---|---|---|---|---|---|---|
| **C** | **S** | **S-M** | **S-F** | **S-N** | **PL** | **PL-M** | **PL-F** | **PL-N** |
| *?a* | *?a-sg-m* | *?a-sg-m* | – | – | *?a-pl-m* | *?a-pl-m* | – | – |

**Table 13.**

There are several possible solutions. The first thing to do in any case is to subdivide the declension schemes further. Following [15], there are more than 26 basic declension schemes. One possibility would be to put them into a huge feature matrix, which would yield more than 26 columns. Obviously, this is not a very elegant solution, for most of the matrices will be extremely sparse. Additionally, the fact that the specific declension schemes can be grouped together and share most of the endings is not taken into account by this solution either.

The possibility chosen here is to add supplementary features which denote the declension scheme to the lexical constructions and to the suffix constructions. It is important to note that no *disjunctive* features are needed here; rather the feature matrix is used together with a *conjunction* of new features. Additionally, the masculine declension scheme is subdivided into three schemes for *virile*, *animate* and *inanimate* and this distinction is introduced into the feature matrix paradigm. However, as mentioned before, this categorization is not always valid, since there are inanimate objects following the animate scheme; therefore, the three schemes are denoted by *M1*, *M2* and *M3*. In total, this results in nested feature matrices consisting of ten feature columns, one case column and two number columns (overall 13 columns).

Another special case is the ending pair *-i* and *-y*, for they mostly mark the same cases. Which one is to be applied depends on the stem: In some declension schemes *-y* is the default ending, and it is substituted by *-i* after *k* and *g*. In other schemes, *-i* is the default ending and has to be exchanged by *i*.

In the remainder of this paper the focus is not to implement all the different schemes, but rather to show how the pursued approach can be used to represent the whole complexity of this intricate declension system. The next section deals with the actual implementation of the system in FCG and explains it step by step.

## 4 Operationalization in FCG

The forthcoming section aims to show how a subset of the Polish grammar can be implemented in FCG and how nested features matrices (further called feature matrices) can be used to facilitate morphological and phrasal agreement. As mentioned in an earlier chapter of this Volume [7], the key idea of construction grammars is to treat every linguistic item as a *form-meaning pair*, a construction. However, not all constructions in FCG have to be form-meaning pairs. Morphological processing can be modeled by exploiting the fact that FCG also allows *form-form pairs*, that is, pure syntactic constructions. The need for syntactic constructions will become obvious when the implementation of morphological processing in the Polish example is explained.

Moreover, the generation of constructions is facilitated by templates which also have been introduced in earlier chapters [7, 9, 10]. In particular the morphological templates from [10] and [9] were adapted to deal with inflection and stem affixes. Additional templates for feature matrix creation were already developed for [11] and were extended for this study to cover nested feature matrices.

The actual application order of the constructions is guided by two factors. On one hand, only appropriate constructions are considered during the application process, since they have to match the current transient structure. In this sense, constructions are looking for and triggered by certain unit features of the current transient structure. One the other hand, the grammar designer can also group constructions into *construction sets* and induce an explicit ordering on the families. In the presented grammar constructions are grouped into the following sets:

**Lexical** constructions provide the basic lexical items.
**Functional** constructions map lexical items to their syntactic function in a phrase or sentence.
**Negative and positive verb** constructions determine if the phrase has a negative sense, expressed by the word *nie*.
**Marked inflection** constructions handle the proper case, number and gender related endings for nouns.
**Unmarked inflection** constructions treat noun forms which do not exhibit an ending.

**Stem** constructions provide the appropriate changes to stems of nouns, according to their stem class and the attached ending.
**Number** constructions determine the right number of a noun.
**Phrasal** constructions take care of phrasal agreement.
**Sentential** constructions add context on the meaning side and punctuation on the form side.

In the following the most important constructions are explained in detail. In order to give the reader an overview of the machinery necessary for the implementation of the formalized linguistic problems, the constructions will be presented both from the design as well as the operational level; that is, template definitions as well as graphical representations of constructions (using the tools presented earlier in [23]) are given.

### 4.1 Feature Matrices

So far, feature matrices were only considered in an abstract way, now their implementation in FCG is presented. In fact, the implementation is rather straightforward, since a matrix can be modeled as a list of lists. Thus, the following code shows the nested feature matrix for the noun *człowiek* (man) (see Table 10) in FCG notation:

```
((nom ?nom (nom-sg ?nom-sg-m1 ?nom-sg-m1 - - - -)
           (nom-pl ?nom-pl-m1 ?nom-pl-m1 - - - -))
 (gen ?gen (gen-sg ?gen-sg-m1 ?gen-sg-m1 - - - -)
           (gen-pl ?gen-pl-m1 ?gen-pl-m1 - - - -))
 (dat ?dat (dat-sg ?dat-sg-m1 ?dat-sg-m1 - - - -)
           (dat-pl ?dat-pl-m1 ?dat-pl-m1 - - - -))
 (acc ?acc (acc-sg ?acc-sg-m1 ?acc-sg-m1 - - - -)
           (acc-pl ?acc-pl-m1 ?acc-pl-m1 - - - -))
 (ins ?ins (ins-sg ?ins-sg-m1 ?ins-sg-m1 - - - -)
           (ins-pl ?ins-pl-m1 ?ins-pl-m1 - - - -))
 (loc ?loc (loc-sg ?loc-sg-m1 ?loc-sg-m1 - - - -)
           (loc-pl ?loc-pl-m1 ?loc-pl-m1 - - - -)))
 (voc ?voc (voc-sg ?voc-sg-m1 ?voc-sg-m1 - - - -)
           (voc-pl ?voc-pl-m1 ?voc-pl-m1 - - - -)))
```

The overall structure is a list which contains one sublist per case. Each case list consists of a symbol denoting the case name (e.g. `nom`), the case (column) variable (`?nom`) and two sublists, one for each number. Again, the number lists contain a symbol for the case-number combination (`nom-sg`), a number (column) variable (in this example the first `?nom-sg-m1`) and finally the actual feature cells. In the example, all entries except the one at the *M1* positions are marked by a '-', since *człowiek* follows the masculine virile declension scheme.

### 4.2 Lexical and Morphological Constructions

One one hand, there are lexical constructions which are simple form-meaning mappings that translate a semantic entity denoting an individual, an object or an event into an appropriate lexical representation. On the other hand morphological constructions deal with the attachment of the right stem and endings to these representations. The focus of this study is on the morphology of nouns. Therefore, first the lexical construction for the noun *dziewczyna* (girl) is given, and it is explained how production and parsing on the lexical level works in FCG. The following example parses the utterance `("dziewczy" "-n" "-a")` and yields its semantic representation `(girl girl-set context)`. It is important to note that the very same constructions can also be used to produce the utterance from the latter semantic representation.

**Semantics** In FCG, semantic representations roughly correspond to second order predicates. However, FCG does not use a formal inference system, so the way of using the predicates is not as strict as in formal predicate calculus. On the semantic side, a girl is presented by the predicate `(girl ?girl-set ?context)`. The first position of the predicate denotes the predicate name, followed by an arbitrary number of arguments. In this case, a helpful interpretation is to consider predicates as functions which calculate output sets or entities from input sets. Hence, the girl predicate calculates the set of girl individuals from a given context set (consisting of different objects and individuals). In a more common predicate calculus representation, the girl predicate might be written as `Girl(`$X$`, `$Y$`)`, where the predicate calculus set variables $X$ and $Y$ correspond to the FCG variables `?girl-set` and `?context`, respectively. In the full FCG grammar solution presented in this paper, each noun is accompanied by another predicate denoting whether only one individual or a set of individuals is referenced, namely `(single-entity ?entity-set)` or `(set ?entity-set)`. A good way to interpret the `single-entity` and `set` predicates is to regard them as additional constraints rather than functions. For example, the following predicate set denotes one specific girl individual in the base set, at least if such an individual exists in the context:

```
(girl ?girl-set ?context) (single-entity ?girl-set)
```

Note how the variable `?girl-set` appears in both predicates ensuring the correct variable binding. The `single-entity` and `set` predicates are introduced by the constructions from the *number* construction set.

**Lexicon** The following code uses lexical templates as presented in [7] to create the *girl* construction, at first without feature matrices:

```
(def-lex-cxn girl-cxn
    (def-lex-skeleton girl-cxn
        :meaning (== (girl ?girl-set ?context))
        :args (?girl-set ?context)
        :string "dziewczy")
    (def-lex-cat girl-cxn
        :sem-cat (==1 (class indiv))
        :syn-cat (==1 (lex-cat noun)
                      (gender feminine))
        :phon-cat (==1 (stem "-n")
                       (stem-class hard)
                       (palatal-plural-endings -)))
```

The resulting construction is depicted in Figure 1. The construction is actualized by a coupled feature structure, consisting of a semantic (left) and a syntactic (right) pole. The upper boxes above the dashed line basically state how the transient feature structure should look in order to be manipulated by this construction. The lower part contains information that should be added to the transient feature structure by this construction in terms of J-Units.

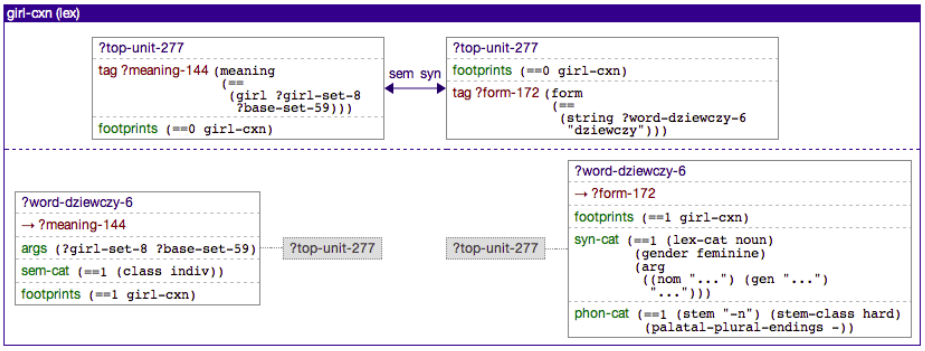


**Fig. 1.** Lexical construction for the (stem of the) noun *dziewczyna* (girl). Due to space constraints, the full feature matrix is omitted.

As visible in the `string` argument of the `def-lex-skeleton`, the lexical item for girl only contains the form `dziewczy`, lacking the stem and a concrete case marking. These must be added by other constructions, considering the intended case and number of the expression. Therefore, the `def-lex-cat` template is used to add more information, particularly about the syntactic and phonetic properties of the word. It assigns the lexical category to be a noun with feminine gender (lines 8 and 9), additionally, it adds phonetic categories (lines 10 to 12): by specifying the default stem and the stem class, only the right endings (in terms of the right constructions) for this noun are considered during further processing.

Finally, the feature `palatal-plural-endings` specifies whether this noun can take palatal plural endings which affect the noun stem. This noun obviously does not, for its nominative plural is *dziewczyn-**y*** and not **dziewczyn-**i***.

After defining the syntactic, semantic and phonetic features, a feature matrix is added to the girl construction. In the first instance, the declension paradigm has to be created which specifies which genders, cases and numbers are available:

```
(defparameter *polish-paradigm*
  (make-matrix-paradigm
   :dimensions ((nom gen dat acc ins loc)
                (sg pl)
                (m1 m2 m3 f n))))
```

Note that the vocative case is left out as it is not be used in this example. The paradigm is an object which is bound to the variable `*polish-paradigm*` and is used throughout the following construction definitions.

Next, the feature matrix is created and added to the girl construction by using the following command:

```
(def-feature-matrix girl-cxn
    :paradigm *polish-paradigm*
    :dimensions (sg-f pl-f)
    :feature (:syn-cat :agr))
```

The crucial parameter is `:dimensions` which specifies which case, number and gender combinations are allowed for this item. Since the noun is feminine, only the feminine singular and the feminine plural columns are set to variables, the other cells are automatically set to '-'. The template takes care of converting the necessary case and number columns into variables or pluses, and looks for the possibility of unifying variables. The `feature` parameter specifies at which exact location in the coupled feature structure the matrix is inserted. In this example, a new feature *agr* (for agreement) containing the feature matrix is created, which is appended to the *syn-cat* feature of the syntactic pole.

**Noun Inflection** At this point, only the constructions for the core of the noun *girl* is available, now the constructions which deal with the endings are shown. The following template creates a construction for the -*a* ending:

```
(def-inflection-affix-cxn inflection-suffix-a
    (def-inflection-affix-skeleton inflection-suffix-a
        :suffix "-a"
        :syn-cat (==1 (lex-cat noun)
                      (gender ?gender))
        :phon-cat (==1 (stem-class ?stem-class))
        :impose-phon-cat (==1 (stem-palatalized -)))
    (def-inflection-affix-feature-matrix
        inflection-suffix-a
          :paradigm *polish-paradigm*
          :feature (:syn-cat :agr)
          :dimensions
           (nom-sg-f
            gen-sg-m1 gen-sg-m2 gen-sg-m3 gen-sg-n
            acc-sg-m1 acc-sg-m2)
          :feature (:syn-cat :agr)))
```

Several new templates arise here, which are adapted to the creation of inflection affixes. Again, there is an overarching template as well as a skeleton and feature matrix template. An important difference is that the resulting construction will not consist of a semantic and a syntactic, but of two syntactic poles. The reason for this is that the ending suffix does not add any new information on the semantic side, but only actualizes the case and number marking. Role assignment and all other semantically relevant tasks are taken care of by phrasal constructions which will be shown later.

The application mechanism for pure syntactic constructions is the same as for normal constructions, that is, it is divided into a matching and a merging phase. However, both matching and merging apply to the syntactic pole of the current transient feature structure.

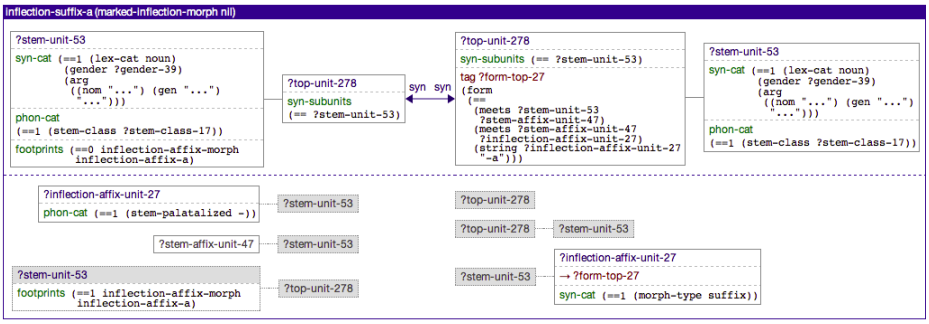


**Fig. 2.** Morphological construction for the *-a* ending. Due to space constraints, the full feature matrices are omitted.

The most important difference is the `suffix` parameter by which a string denoting the actual ending is passed. Alternatively, the parameter `infix` may be used which also takes a string as its value. This is necessary since there exist cases which are in principle unmarked, but which introduce a gap vowel before the stem. For example, the genitive plural of the feminine noun *deska* (board) is *desek*, that is, an *e* is introduced before the stem consonant *k*. This phenomenon can be handled if the *e* is treated like a normal inflection ending, but which occurs in the infix instead of suffix position.

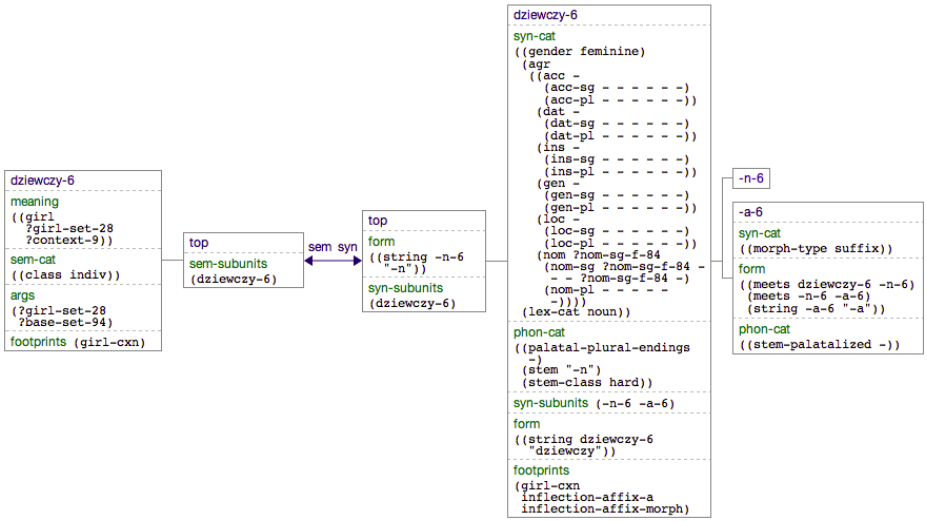


**Fig. 3.** Transient structure resulting after the application of *girl-cxn* and *inflection-suffix-a-cxn*.

Another important new parameter is `impose-phon-cat`. Intuitively speaking, the difference between this parameter and `phon-cat` is that the latter formulates constraints concerning the transient structure before application of the construction. That means that any feature present in the `phon-cat` of the construction must also be present in the `phon-cat` of the transient structure – otherwise the construction will not apply. On the other hand, the `impose-phon-cat` in the example *adds* the feature `(stem-palatalized -)` to the transient structure after application, and therefore affects constructions applying afterwards. In particular, this feature declares that with this ending no palatalization of the stem occurs, and therefore the default stem has to be attached.

The *-a* ending construction depicted in Figure 2 and the resulting transient structure after the application of *girl-cxn* and *inflection-suffix-a-cxn* shown in Figure 3 illustrate the parsing and production process. For a syntactic ending construction only the right pole of the transient structure has to be considered. The *inflection-suffix-a-cxn* attaches a new subunit to the noun unit. In the con-

struction, the noun unit is called `?stem-unit-73`, that is, its name is a variable. Therefore, it can be bound to the `dziewczy-6` unit in the transient structure. Similarly, `?inflection-unit-33` transforms to `-a-6`. The `-a-6` unit contains the feature `(morph-type suffix)` in its `syn-cat` which determines that the stem added in the next step must be an infix. Note that the feature matrix in the `syn-cat` of `dziewczy-6` has already determined the nominative singular case to be the right solution after application of the *inflection-suffix-a-cxn*. Furthermore, the *inflection-suffix-a-cxn* adds another empty unit named `-n-6`, which functions as a placeholder for the stem added by the construction presented in the next section.

Before turning to stem constructions, it should be mentioned how unmarked cases are handled. Unmarked forms can by actualized by the same inflection templates as for marked forms, but no `suffix` or `infix` arguments are passed. However, also in the case of unmarked forms an inflection-unit is added to the transient structure. This unit is necessary since stem constructions need information whether or not they are affected by palatalization. The stem constructions expect this information to be located in an inflection unit. Therefore, the difference to marked forms is that the inflection unit for null endings does not contain any `string` feature. Furthermore, unmarked inflection constructions are grouped into another construction set, since they have to be applied after the constructions that deal with marked endings. Otherwise, unmarked inflection constructions would apply even if there actually is an ending.

**Stems** As explained before, palatalization has an effect on the stem of a noun. Stem constructions are very similar to inflection constructions and also use structurally related templates. The following template creates the stem infix *-n*:

```
(def-stem-affix-cxn stem-infix-n
    (def-stem-affix-skeleton stem-infix-n
        :infix "-n"
        :syn-cat (==1 (lex-cat noun)
                      (gender ?gender))
        :phon-cat (==1 (stem "-n") (stem-class hard))
        :inflection-phon-cat  (==1 (stem-palatalized -))))
```

Most of the parameters contain the same function as the ones in the inflection template. An important new parameter is `inflection-phon-cat` which is the counterpart of the `impose-phon-cat` from the inflection construction. In this example, it states that the inflection attached to the noun must not palatalize the stem; otherwise this construction would not be applied and a different one would have to be chosen. If needed, a feature matrix can be added to the stem as well, which is not necessary in this case because the *-n* stem occurs in all cases. Figure 4 shows the resulting construction.

After parsing `("dziewczy" "-n" "-a")` or producing `(girl girl-set context)` the `-n-6` unit on the syntactic pole now contains the actual string representation

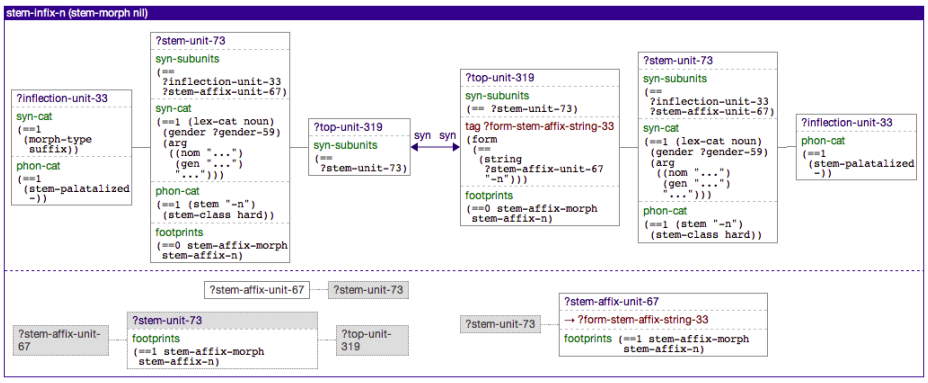


**Fig. 4.** Morphological construction for the *-n* stem.

for the stem. The syntactic pole of the obtained coupled feature structure are depicted in Figure 5.

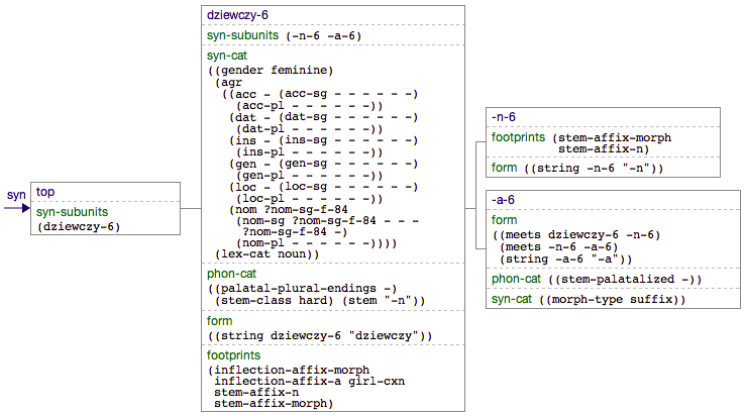


**Fig. 5.** Syntactic pole of the coupled feature structure after the application of lexical and morphological constructions.

**Number** Now that all work on the syntactic side is done, a predicate denoting the right number of individuals must be introduced on the semantic side. There are two specific constructions, *singular-cxn* and *plural-cxn*. For the sake of brevity, the constructions are not depicted here, also because they do a fairly simple job: in the semantic pole, they introduce the `single-entity` or `set` pred-

icate, respectively. In order to do so, each of them includes a feature matrix in the syntactic pole which either contains variables in all the singular or in the plural cells. In terms of the previous example, the *singular-cxn* should apply because the feature matrix in the `dziewczy-unit` in Figure 5 determines the case to be nominative singular.

### 4.3 Phrases

In the following paragraphs a more complex example in terms of a simple sentence will be developed. Only the SVO pattern considered, therefore, the relatively free word order of Polish is not accounted for. However, this can be actualized by using a field topology approach which was implemented in FCG for German as presented in a later chapter of this volume [24]. Moreover, only transitive verbs are considered and no verb morphology is modeled. Therefore, all verbs are in the third person singular or infinitives. However, morphology for verbs can be added in a similar way as the noun morphology presented in the preceding sections. The grammar can be also easily extended with more complex phrases by adding constructions for more complex patterns similar to the ones presented in the forthcoming explanation.

**Verbs** First, constructions for verb forms are added. The following lexical construction defines the verb form *widzi* (he/she/it sees):

```
(def-lex-cxn see-cxn
    (def-lex-skeleton see-cxn
        :meaning (== (see ?see-set ?base-set)
                     (seer ?see-set ?seer)
                     (seee ?see-set ?seee))
        :args (?see-set ?seer ?seee ?base-set)
        :string "widzi")
    (def-lex-cat see-cxn
        :sem-cat (==1 (class event)
                      (sem-val
                       (==1 (agent ?see-set ?seer)
                            (patient ?see-set ?seee))))
        :syn-cat (==1 (lex-cat verb)
                      (verb-form inflected)))
    (def-feature-matrix see-cxn
        :paradigm *polish-paradigm*
        :dimensions (nom)
        :feature (:syn-cat :subject-agr))
    (def-feature-matrix see-cxn
        :paradigm *polish-paradigm*
        :dimensions (acc gen)
        :feature (:syn-cat :direct-object-agr)))
```

This construction is very similar to the lexical noun constructions presented before, only the semantic features are different because the verbs considered here describe events rather than individuals. Notice that the construction contains two feature matrices, one for agreement with the subject, another for agreement with its direct object. The subject is forced to take the nominative case, while the direct object can take either the accusative or genitive case – owed to the genitive of negation. In order to distinguish the two feature matrices, the feature matrix corresponding to the subject is located in the `subject-agr` feature, while the other one is located in the `direct-object-agr` feature.

The constructions for verbs which do not take the accusative look almost the same, such as *macha* (he/she/it waves), which takes the instrumental case. The main difference is that they only allow the instrumental case in the direct object agreement feature matrix, since they are not affected by the genitive of negation.

Since verb morphology is not covered in this paper a new construction for every verb form has to be defined. However, this is not too much work since only infinitives and verbs in the third person singular are considered. The `see-infinitive-cxn` looks almost the same as the `see-cxn` construction, except for having the `verb-form` feature, which is set to `infinitive`. Note that this means that there is no difference on the meaning side between an inflected and an uninflected verb construction. As shown later, this can become a problem in production which is overcome by the introduction of sentential constructions.

Beside the full verbs also inflected forms of auxiliaries like *chce* (he/she/it wants) are defined. In order to distinguish these verbs, the auxiliaries take the additional `(verb-type auxiliary)` feature in their `syn-cat`.

**Negation** In order to handle the genitive of negation, in the first instance, another lexical construction for the negation marker *nie* (not) is needed. It is fairly trivial, for it translates this *nie* string into the predicate `(not ?event)`. The argument of the `not` predicate is an event which is introduced in the meaning of the verbal construction above.

```
(def-lex-cxn not-cxn
    (def-lex-skeleton not-cxn
        :meaning (== (not ?event))
        :args (?event)
        :string "nie")
    (def-lex-cat not-cxn
        :syn-cat (==1 (lex-cat particle))
        :sem-cat (==1 (class modifier))))
```

In terms of FCG hierarchy [25], it also adds a new unit to the transient structure which represents this negation marker or predicate, respectively. Note that only sentential negation is considered in this example, that is, no constituent negation. This type of negation also exists in Polish and is marked by the same word *nie*.

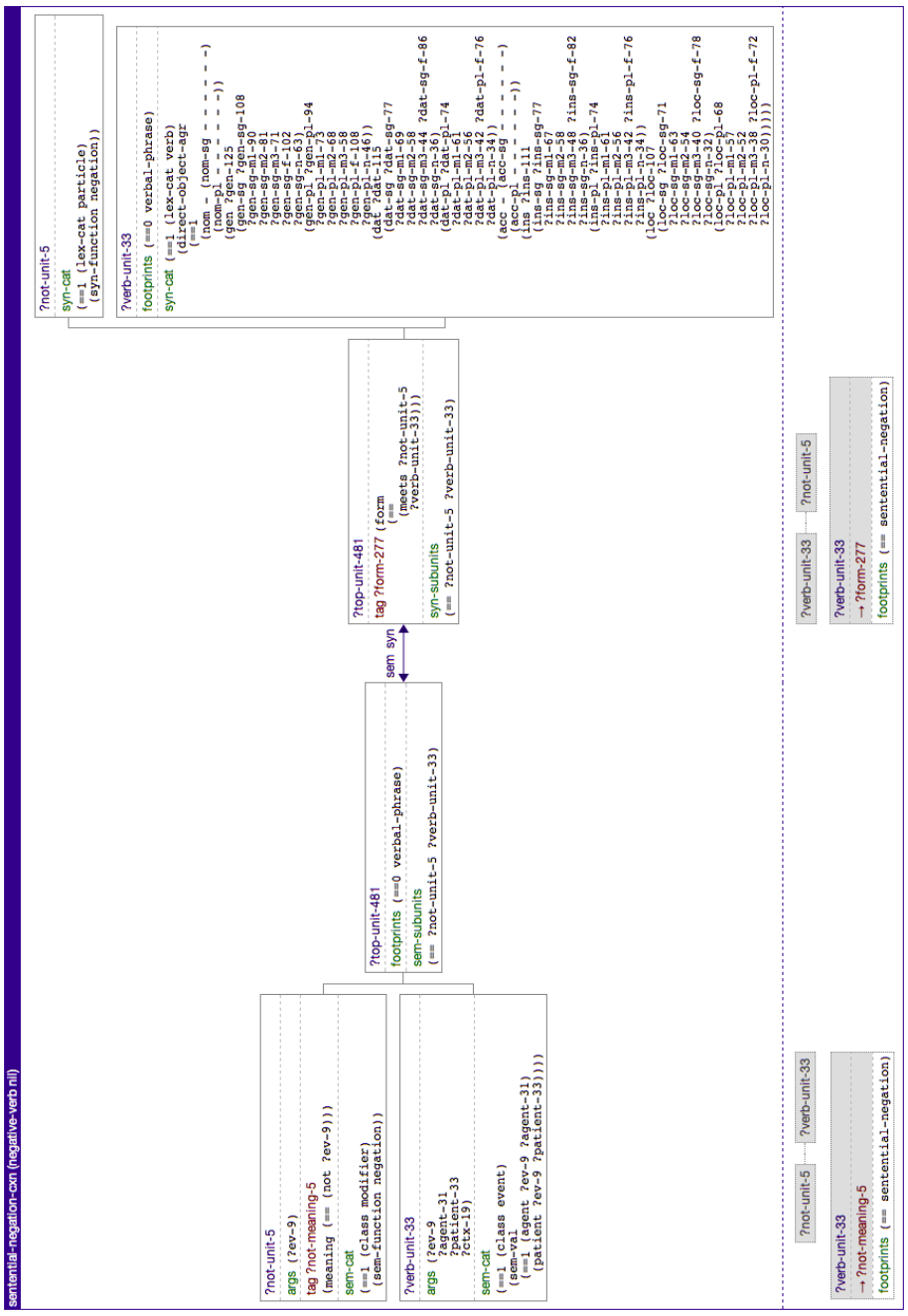


**Fig. 6.** The *sentential-negation-cxn* prevents the direct object of the verb construction from taking the accusative case if the negation marker *not* is present.

In the next step, a construction is needed which propagates the appearance of the negation marker to the verbal unit. It must bind the `?event` variable in the *not-cxn* to the verbal unit, and must also affect the direct object agreement feature matrix of a verbal unit. More precisely, it sets the accusative row of this feature matrix to '-'. The nominative row is also set to '-', since direct objects of transitive verbs do not take this case either. The *sentential-negation-cxn* is depicted in Figure 6.

However, a construction which discovers negation is not sufficient, but also a construction for the positive case is necessary. The reason for this is that there must be a way to sort out the possibility that the direct object of a verb like *widzieć* (see) can take the genitive case, if the phrase is not negated. This *positive-verb-cxn* is much simpler than its negative counterpart, for it does not introduce any new units, but just sets the genitive row of the direct object feature matrix of the affected verb unit to '-'.

Note that these constructions are also able to deal with verbs which always require their direct object to be in the genitive case, that is, also if no negation is present. An example of such a verb is *szukać* (search). In this case neither the *sentential-negation-cxn* nor the *positive-verb-cxn* apply, for the feature matrix for the direct object of *szukać* contains a '+' in the genitive row. The *sentential-negation-cxn* does not apply because no negation marker is present, the *positive-verb-cxn* cannot apply, for its feature matrix contains a '-' in the genitive row. Hence, their feature matrices cannot be unified and no change is made to the verbal unit.

**Functional Constructions** Before the units created by the lexical constructions can be processed by phrasal constructions, the functional role of the parts of speech has to be determined. This is the task of the functional constructions which are created by the `def-fun-cxn` template. They are fairly simple, therefore only one exemplary construction which assigns the predicate role to an inflected verb is given here:

```
(def-fun-cxn predicate-cxn
    :sem-function action
    :sem-cat (==1 (class event))
    :syn-cat (==1 (lex-cat verb)
                  (verb-form inflected))
    :syn-function predicate)
```

**Verbal Phrases** In order to model the long distance genitive of negation a construction is created that allows auxiliary verbs and infinitives to be grouped into a verbal phrase. The construction is depicted in Figure 7. In order to apply, it requires an `auxiliary-unit` and an `infinitive-unit`.

It is important to see how long distance dependencies get propagated. During the application of the construction the feature matrices of the `infinitive-unit` are moved into a new `auxiliary-infinitive-verbal-phrase-unit` which then

heads the verbal units. This new unit behaves as if it were a verb, and the *sentential-negation-cxn* can apply to it.

In a similar way also the processing of a chain of infinitives could be implemented: Pairs of infinitives are headed by a new phrasal unit, and the last infinitive's feature matrices are moved to this phrasal unit.

**Sentential Phrases** In the last step, phrasal constructions are used to arrange all the parts of speech with transitive phrase constructions. Since they are quite complex constructions, only the templates for creating feature matrices and feature matrix agreement are presented. For more information on how phrasal agreement is realized in FCG by this templates, see also [26].

Figure 8 shows the general structure of a *transitive-phrase-cxn*. The construction requires three units, namely a *subject-unit*, an *object-unit* and a *verb-unit*. During application, it introduces a new unit called *transitive-phrase-unit*, which subsumes the aforementioned units. The construction's main task is to assign the right cases to its constituents. In fact, several constructions are needed, more precisely, one for each possible case that a direct object can take. This is necessary because the construction should assign a '+' in one of the case rows and set all other rows to '-'. This is especially needed in production, where the phrasal constructions precede the morphological constructions, which must be able to select the right ending without ambiguity[3].

The following template adds the right feature matrices to the *accusative-transitive-phrase-cxn*:

```
(def-phrasal-feature-matrix
  accusative-transitive-phrase-cxn
    :paradigm *polish-paradigm*
    (?subject-unit
            (:feature (:syn-cat :agr)
             :dimensions (nom)))
    (?object-unit
            (:feature (:syn-cat :agr)
             :dimensions (acc)))
    (?verb-unit
            (:feature (:syn-cat :subject-agr)
              :dimensions (nom))
            (:feature (:syn-cat :direct-object-agr)
              :dimensions (acc))
```

In total, this template creates four feature matrices in this construction: one that will match with the `subject-unit`'s feature matrix, one for the `object-unit`

[3] Another possibility would be to use a phrasal construction containing a feature matrix, forcing one row to be '+' and all others to be '-' but changing the case name (e.g. `nom`) of all the rows to variables. This solution is not pursued here, since it is computationally very expensive.

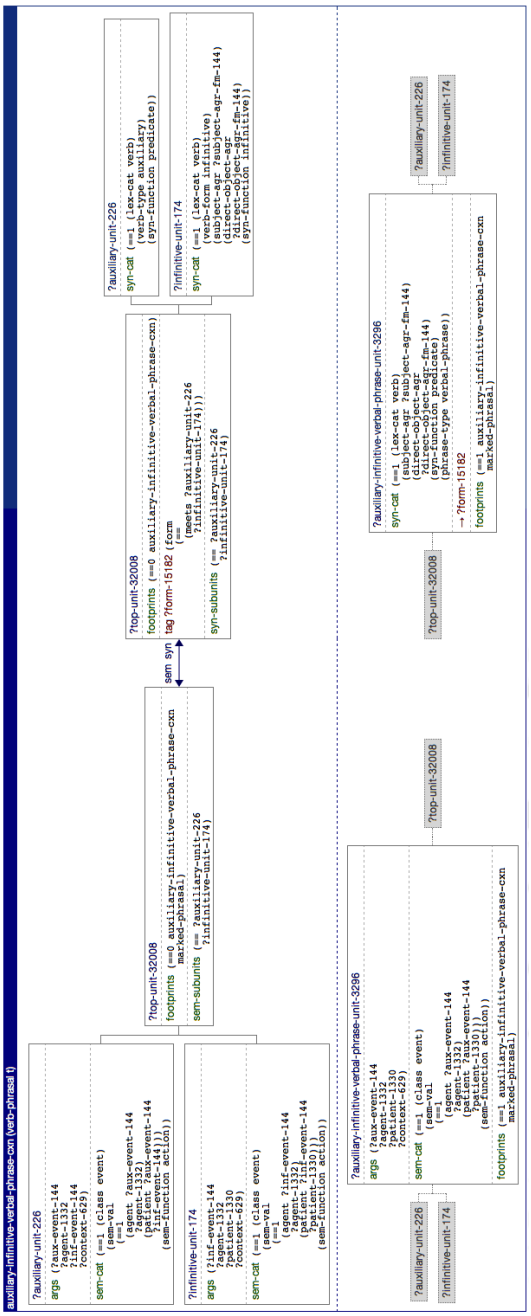


**Fig. 7.** The *auxiliary-infinitive-verbal-phrase-cxn* groups an inflected auxiliary verb and an infinitive.

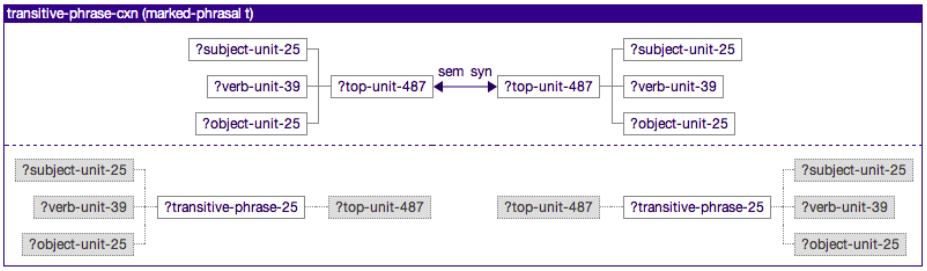


**Fig. 8.** General structure of the *transitive-phrase-cxn*.

and two for the agreement matrices of the `verb-unit`. However, this is still insufficient, since the verb has to agree with both the subject and the object. Therefore, the verb's subject feature matrix must unify with the subject's feature matrix and the verb's object feature matrix must unify with the object's feature matrix. For this purpose, the `def-phrasal-feature-matrix-agreement` template is needed. The following code unifies the direct object feature matrix of the `verb-unit` with the feature matrix of the `object-unit`:

```
(def-phrasal-feature-matrix-agreement
  accusative-transitive-phrase-cxn
    :paradigm *polish-paradigm*
    :agreement
        ((?verb-unit
            :feature (:syn-cat :direct-object-agr))
         (?object-unit
            :feature (:syn-cat :agr))))
```

Similarly, the subject-verb agreement can also be established.

When the construction applies, the units feature matrices unify with the matrices of the phrasal construction. Because of the pairing of the subject-verb and object-verb feature matrices, the right cases are propagated to the units.

### 4.4 Sentence Parsing and Production

Figure 9 shows the transient structure after parsing the following sentence:

`"Micha" "-ł" "nie" "chce" "widzieć" "dziewczy" "-n" "-y" "."`.
(*Michael does not want to see the girl.*)

The following semantical representation is parsed from this sentence:

```
(michal michal-indiv-1 context-1)
```

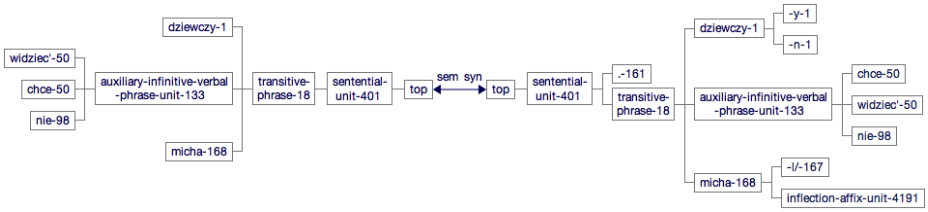


**Fig. 9.** Transient structure resulting after parsing the sentence *Michael does not want to see the girl*.

```
(single-entity michal-indiv-1)
(girl girl-indiv-1 context-1)
(single-entity michal-indiv-1)
(not want-ev-1)
(want want-ev-1 context-1)
(wanter want-ev-1 michal-indiv-1)
(wantee want-ev-1 see-ev-1)
(see see-ev-1 context-1)
(seer see-ev-1 michal-indiv-1)
(seee see-ev-1 girl-indiv-1)
(context context-1))
```

In order to see the result of the feature matrix agreement, the syntactic `dziewczy-unit` is shown in more detail in Figure 10. As can be clearly seen, the feature matrix determines that the noun takes the genitive singular case.

The *sentence-cxn*, not yet explicitly mentioned so far, adds the `(context context-1)` predicate on the semantic and the full stop on the syntactic side. It applies if all units have been subsumed under a phrasal unit and ensures that no units are uncovered. This might happen in production since the semantic poles of the inflected and the infinitive constructions look exactly the same – their meanings are identical, but their form realizations are different. Therefore, FCG considers the possibility of using the inflected verb form construction for both *want* and *see*. Of course, two inflected verbs forms cannot be grouped together by the *auxiliary-infinitival-verbal-phrase-cxn*, and therefore the following defective sentence is produced:

*"Michał" "nie" "widzi" "dziewczyny" "chce".
(**Michael does not see the girl, wants.*)

However, the sentence construction will not apply in this case and leave the *(context context-1)* predicate uncovered. This will signal FCG that it should look for a better solution that processes all the meaning predicates.

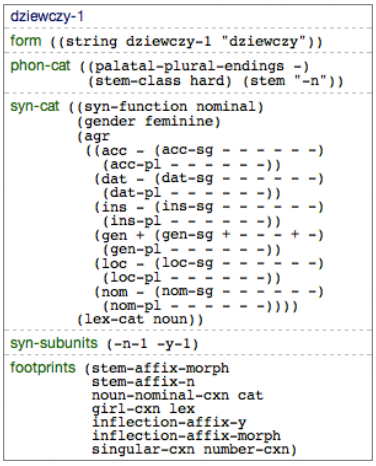


**Fig. 10.** The syntactic `dziewczy-unit` resulting after parsing the sentence *Michael does not want to see the girl.*

Note that multiple subclauses do not pose a problem in principle. In order to actualize them, several slightly differing sentence constructions that require a finite amount of subphrases could be used in order to group the subphrases in an appropriate way.

## 5 Conclusion

This paper has explained how a complex declension system can be operationalized in FCG. Polish was chosen as a representative from the group of Slavic languages in order to verify if FCG can cope with highly irregular natural language examples. Feature matrices have proven to be an elegant representation for this system. They can be used to model agreement from the morphological to the phrasal level. The key idea is to include the feature matrices in constructions for nouns, morphological suffixes and phrases. Each of the constructions contains a feature matrix encoding in which the possible cases, numbers and genders of the item are encoded. Indeed, different constructions for the same ending might be necessary, if they are required for the disambiguation of declension variations within the same gender. Which variation a noun must exhibit in order to be combined with an ending is stored in supplementary features which are added to the constructions of the noun and the ending. Moreover, the endings contain more features which encode their effect on the stem. This allows palatalization to be modeled. The presented approach can also deal with unmarked cases. In some of these unmarked cases, a gap vowel has to be introduced before the stem consonant. This special case can be solved by treating the gap vowel as an infix

marking. In the same way as palatalizing endings, a gap vowel is indicated by adding supplementary features to the construction that represents the gap vowel. This way, the right constructions are triggered in order to parse and produce a noun form in a grammatically and syntactically correct manner.

Furthermore, a simple example of the long distance genitive of negation has been implemented. A special focus has been on the case where the genitive of negation has an effect on the direct object of a nested infinitive when the predicate is negated. Phrasal constructions subsume the predicate and the infinitive in a new unit and propagate information about verbal agreement up to this unit. This allows for the right constructions to apply and to change the feature matrices representing the verbal agreement correctly.

The presented grammar can be extended in several ways. For instance, the grammar can be scaled up by adding adjectives and pronouns. Morphologically, these grammatical items can be treated in an analogue way to the nouns, that is, every item will exhibit its own feature matrix and specific endings. From a syntactic point of view, more flexible phrasal constructions will be necessary. However, it has already been extensively shown how these different grammatical items can be handled in FCG (see this volume and also [1]).

More work on how to handle long distance dependencies in FCG in general could be done. Polish exhibits a strong negative concord, that is, a negative particle such as *nikt* (nobody), which always requires the negation marker *nie*, despite not expressing double negation [27]. This phenomenon calls for a general way to represent long distances, and future work should concentrate on how these dependencies can be modeled in FCG.

## Acknowledgements

This research was conducted at the Sony Computer Science Laboratory in Paris and at the University of Brussels (VUB AI Lab). Funding has come from the Sony Computer Science Laboratory as well as from the European research project ALEAR (FP7, ICT-214856). I would like to thank all members of the language groups at Sony CSL and at the VUB AI Lab for insightful discussions and helpful comments, and Liviu Ciortuz for pointing out relationships of my work to other work in the Slavic linguist community.

## Bibliography

[1] Steels, L., ed.: Design Patterns in Fluid Construction Grammar. John Benjamins, Amsterdam (2011)
[2] Steels, L., ed.: Computational Issues in Fluid Construction Grammar. Springer Verlag, Berlin (2012)
[3] Steels, L., ed.: Experiments in Cultural Language Evolution. John Benjamins, Amsterdam (2012)
[4] Przepiórkowski, A., Kupść, A.: Why formal grammar? (1999)

[5] Borsley, R.D., Przepiórkowski, A.: Slavic in Head-Driven Phrase Structure Grammar. CSLI Publications, Stanford, CA (1999)
[6] Pollard, C., Sag, I.A.: Information-based syntax and semantics: Vol. 1: fundamentals. Center for the Study of Language and Information, Stanford, CA, USA (1988)
[7] Steels, L.: Design methods for Fluid Construction Grammar. In Steels, L., ed.: Computational Issues in Fluid Construction Grammar. Springer Verlag, Berlin (2012)
[8] Beuls, K.: Construction sets and unmarked forms: A case study for Hungarian verbal agreement. In Steels, L., ed.: Design Patterns in Fluid Construction Grammar. John Benjamins, Amsterdam (2011)
[9] Beuls, K.: Handling scope in Fluid Construction Grammar: A case study for Spanish modals. In Steels, L., ed.: Computational Issues in Fluid Construction Grammar. Springer Verlag, Berlin (2012)
[10] Gerasymova, K.: Expressing grammatical meaning with morphology: A case study for Russian aspect. In Steels, L., ed.: Computational Issues in Fluid Construction Grammar. Springer Verlag, Berlin (2012)
[11] van Trijp, R.: Feature matrices and agreement: A case study for German case. In Steels, L., ed.: Design Patterns in Fluid Construction Grammar. John Benjamins, Amsterdam (2011)
[12] Ciortuz, L., Saveluc, V.: Fluid Construction Grammar and feature constraints logics. In Steels, L., ed.: Computational Issues in Fluid Construction Grammar. Springer Verlag, Berlin (2012)
[13] Ciortuz, L.: LIGHT — A constraint language and compiler system for typed-unification grammars. In: KI-2002: Advances in Artificial Intelligence. Volume 2479., Berlin, Germany, Springer-Verlag (2002) 3–17
[14] Ciortuz, L., Saveluc, V.: Learning to unlearn in lattices of concepts: A case study in Fluid Construction Grammars. In: Proceedings of SYNASC 2011, Timişoara, Romania, IEEE Computer Society (2011) 160–167
[15] Orzechowska, A.: Rzeczownik. In Grzegorczykowa, R., R.L., Wróbel, H., eds.: Gramatyka współczesnego języka polskiego: Morfologia [A Grammar of Contemporary Polish: Morphology]. Volume 1. Państwowe Wydawnictwo Naukowe, Warszawa (1998)
[16] Dąbrowska, E.: Learning a morphological system without a default: the polish genitive. Journal of Child Language (28) (2001) 545–574
[17] Kowalik, K.: Morfonologia. In Grzegorczykowa, R., R.L., Wróbel, H., eds.: Gramatyka współczesnego języka polskiego: Morfologia [A Grammar of Contemporary Polish: Morphology]. Volume 1. Państwowe Wydawnictwo Naukowe, Warszawa (1998)
[18] Buttler D., Kurkowska H., S.H.: Kultura języka polskiego. Zagadnienia poprawności gramatycznej [Culture of Polish language: Issues on grammatical correctness]. Państwowe Wydawnictwo Naukowe, Warszawa (1973)
[19] Przepiórkowski, A.: Long distance genitive of negation in polish (2000)
[20] Flickinger, D.P.: On building a more efficient grammar by exploiting types. Natural Language Engineering **6**(1) (2000) 15–28

[21] Ramsay, A.: Disjunction without tears. Computational Linguistics **16**(3) (1990) 171–174
[22] De Beule, J.: A formal deconstruction of Fluid Construction Grammar. In Steels, L., ed.: Computational Issues in Fluid Construction Grammar. Springer Verlag, Berlin (2012)
[23] Loetzsch, M.: Tools for grammar engineering. In Steels, L., ed.: Computational Issues in Fluid Construction Grammar. Springer Verlag, Berlin (2012)
[24] Micelli, V.: Field topology and information structure: A case study for German constituent order. In Steels, L., ed.: Computational Issues in Fluid Construction Grammar. Springer Verlag, Berlin (2012)
[25] De Beule, J., Steels, L.: Hierarchy in fluid construction grammar. In Furbach, U., ed.: Proceedings of the 28th Annual German Conference on Artificial Intelligence. Volume 3698 of Lecture Notes in Artificial Intelligence., Berlin, Germany, Springer Verlag (2005) 1–15
[26] Steels, L.: A design pattern for phrasal constructions. In Steels, L., ed.: Design Patterns in Fluid Construction Grammar. John Benjamins, Amsterdam (2011)
[27] Richter, F., Sailer, M. In: Negative Concord in Polish. CSLI Publications, Stanford, CA (1999)