Unification Morphology Grammar

Software System
for Multilanguage Morphological Analysis

Jan Hajič
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University
Malostranské nám. 25
CZ-118 00 Praha 1
tel: +42-2-24510286, ext. 256
fax: +42-2-532742
e-mail: Jan.Hajic@ff.cuni.cz
PhD Thesis

I-3 Mathematical Linguistics

Prague, May 29th, 1994

Tato práce by nevznikla, kdybych nepřipravil o mnoho času a her svoje dvě děti; nevznikla by, kdybych neměl podporu a zázemí u své ženy; nevznikla by, nebýt životního příkladu mé matky, která mě k počítačové lingvistice přivedla a která mi mnohokrát podala pomocnou ruku nejen v pracovních věcech (ačkoli, sebekriticky přiznávám, málokdy jsem jí dokázal dát najevo, jak si této pomoci vážím), nevznikla by, pokud by mě můj školitel, prof. Petr Sgall, nenasměroval správným směrem, ani kdyby nedržel pevnou ruku nad jejím průběhem; nevznikla by, nebýt odvahy Maghi King z ženevského ISSCO vzít mě "na zkušenou" do svého institutu; nevznikla by, nebýt pochopení mého bývalého zaměstnavatele (VÚMS Praha) v době, kdy to nebývalo zvykem; nevznikla by ani, pokud by za mě často nezaskočili v mnoha věcech moji spolupracovníci; a nevznikla by ani nebýt mnoha dalších lidí, mně blízkých i vzdálenějších, kteří mi, přímo či nepřímo, pomohli udržet tempo, elán a optimismus radou, pomocí i milým slovem.

Contents

Chapter 1

Introduction and Background

On-line corpora are increasingly popular as a tool for linguistic research, or, for those using information-theoretic approach, as a direct source of data. However, such corpora are usually available only in the original text form, with little or no processing done on them. Whoever uses such corpora has to first preprocess them to be able to use them as the starting point for his/her experiments. Many researchers thus unnecessarily repeat the same job: text "cleaning", markup, morphology, tagging. It is believed that we can agree on the definitions and format of the output of such preprocessing, and therefore, build tools which would help to reduce, or perhaps even eliminate in some cases, the duplicative and tedious work done on machine readable data by many today. The work presented here forms part of BCP, the Bilingual Concordancy Program developed at ISSCO (Warwick, Hajič and Russell 1990; Warwick-Armstrong et al. 1992).

Morphology of natural languages has been in the focus of computational linguists in non-English speaking countries since the beginnings of natural language processing. Worldwide attention has been turned to it after 1983, when Kimmo Koskenniemi published his infamous thesis (Koskenniemi 1983). Today, even when the questions of time and space are becoming much less serious constraints given the progress in hardware resources, it is necessary to have some serious and wide-coverage description of a morphology system for many languages. In this thesis, a system general enough to be applicable to many different languages exhibiting various linguistics phenomena is presented, with special attention to (highly) inflectional languages, such as Czech and other slavonic languages.

Several software systems designed for automatic processing of Czech have been developed in the past. Many of them encapsulate morphological processing, either of a general nature ((Bémová and Králíková 1988), (Weisheitelová, Králíková and Sgall 1982), (Panevová 1981), (Weisheitelová and Bémová 1979), (Pala 1992)) or suited to a particular system ((Kirschner 1983), (Kirschner 1982a), (Cejpek 1982), (Kirschner 1982b), (Hajič 1987a), (Hajič 1984a), (Panevová and Králíková 1990), (Hajič 1988a), (Hajič and Drózd 1990), (Laciga 1993)). These systems have been either limited in scope, or in coverage (with the recent exception of (Pala 1992)).

Thus the author used the opportunity offered when working as a visiting scientist at ISSCO, Genève, Switzerland in 1990, to start a software project the goal of which has been to provide a tool for morphological analysis of the official Swiss languages (German, French, Italian, with the obvious extension to English), and designed a system capable of easily dealing with the more inflective languages as well (in fact, one inflectional language, such as Czech, contains all the morphological phenomena found in the other more or less analytical languages, but no more). The development of the software continued in Prague and has been finished in 1991, again at ISSCO. The problem of acquiring the data for Czech had already been solved by author's previous work on a spelling-checker (Hajič and Drózd 1990) and a lemmatizer (Laciga 1993), as well as by the work of the other members of the current Institute of Formal and Applied Linguistics. The formal conversion of the lexical data (sometimes meaning some extra handwork, of course) has not been that difficult, although quite time consuming; the initial work done by the author is being followed by Hana Skoumalová, who added the bulk of dictionary entries to increase the coverage of the Czech system. She is currently also converting the Czech system described in this thesis to suit the project MATRACE (Hajič et al. 1992), (Hajičová 1993).

Ideas from many other sources have been used or taken into account during the development of the system. First of all, the basic framework has been shaped by the approach of the Prague linguistic group, as expressed e.g. in (Sgall, Hajičová and Panevová 1986) or (Sgall et al. 1986). The Czech declension system has been influenced primarily by the work of (Sgall 1960) and (Sgall 1967), although it was substantially modified. Furthermore, the present grammar books have been heavily consulted: (Bělič 1981), (Havránek and Jedlička 1963), (Dokulil et al. 1986), (Petr 1986), (Šmilauer 1969) and (Daneš, Grepl and Hlavsa 1987); for the moment, the recent changes in Czech

orthography as expressed in (Martincová 1993) have been disregarded, due to the hesitations whether this to-be-norm is to be taken seriously. Many features of the previous morphological systems developed in Prague have been taken into account, mainly from (Bémová and Králíková 1988), (Weisheitelová, Králíková and Sgall 1982), (Panevová, Cejpek and Zvelebil 1988), (Oliva 1983), and (Panevová 1981).

In the area of morphological formalisms, the following books and articles have been relevant and critically consulted: (Borin 1991), (Finkler and Neumann 1988), (Gazdar 1985), (Kay 1987), (Karlsson 1986), (Karlsson 1985), (Ritchie 1992), (Sågvall 1973), and (Trost 1990).

The main source for the Czech dictionary was a large volume of text provided by different sources, partially machine readable, and the following dictionaries: (Slavíčková 1975), (Filipec and Daneš 1978), (Poldauf 1986), (Havránek 1989), (Těšitelová, Petr and Králík 1985) and (Těšitelová, Petr and Králík 1986).

Inspiration for the linguistic part of the work has been taken also from (Sgall and Weisheitelová 1968), (Sgall and Zima 1986), (Machová 1977), (Panevová 1979), (Trnková 1983); from foreign sources mainly (Klein 1988), (Sproat 1992), (Hellberg 1974), (Matthews 1982), (Booij and van Marle 1993), and (Zwicky and Wallace 1984).

The lexical issues connected with the development of any morphological system have been consulted primarily in (Karlsson and Koskenniemi 1985), (Ritchie et al. 1992), and (Čermák, Králík and Pala 1992).

There is also one large software system using Czech morphological processing, namely the one developed at the Masaryk University in Brno: (Pala 1992), (Osolsobě and Pala 1990), and (Osolsobě, Pala and Franc 1987); although this system has not been known to the author in detail, discussions with the colleagues from Brno have helped, too.

Important statistical data can be found in the works of the former group of statistical linguistics at the Czechoslovak Academy of Sciences: (Těšitelová 1987), (Těšitelová 1992), (Těšitelová 1985), (Těšitelová 1983).

Last but not least, any programming cannot be done without occasionally consulting the basic programming reference – such as (Knuth 1969). The formal languages stuff is best described in (Aho, Hopcroft and Ullman 1976), (Aho and Ullman 1972), (Salomaa 1973) or (in Czech) (Chytil 1984).

Also, author's previous experience in work on various NLP systems found its way into the present system: (Hajič 1984b), (Hajič 1980), (Hajič and

Hajičová 1980), (Jirků and Hajič 1982), (Hajič 1983), (Sgall and Hajič 1984), (Borota and Hajič 1984), (Hajič 1985), (Hajič 1986), (Hajič, Kirschner and Rosen 1986), (Hajič and Zvelebil 1986), (Hajič and Oliva 1986), (Hajič 1987b), (Hajič 1987a), (Hajič 1987c), (Hajič 1988b), (Hajič 1988c), (Hajič 1989a), (Hajič 1989b), (Hajič, Rosen and Hajičová 1992), (Drózd and Hajič 1990), (Brown et al. 1993).

Some parts of this thesis have already been published: (Hajič 1993) – Chapters 2 and 3, (Hajič 1992) – Section 5.1. These parts has been only slightly modified here. The rest consists of yet unpublished work.

Chapter 2

Markup and Basic Processing

Before any meaningful processing can begin, it is necessary to identify word form units, sentence boundaries, numbers (and possibly normalize them), punctuation, and get rid of the remaining garbage. In accordance with the current trend in standardization, the result is marked using SGML, though the Data Type Definition is, obviously, pretty simple.

An example:

- Input Text:

  ... in der Ertragsbilanz gegenüber dem Vorjahr um 2,8 Mrd Fr. ...

- Output Text:

```
...
<f>in
<f>der
<f cap>Ertragsbilanz
<f>gegenüber
<f>dem
<f cap>Vorjahr
<f>um
<w num.orig>2,8
<f num.generated>2.8
<f cap>Mrd
<f cap.abbr>Fr.
...
```

2.1 Word unit identification

Originally, we tried to develop a sophisticated system of handling all the difficult cases of fullstops, other punctuation, hyphens, apostrophes, etc., but it became just a big mess. The problem is a circular one: to decide what a certain apostrophe means, we would have to do some analysis, but for the analysis, we have to know what to analyze... Therefore, we decided to make it simple: we define "words" as being separated by blanks and punctuation only:

self-governing → `<f>self-governing`
isn't → `<f>isn't`

2.2 Number normalization

Numbers are a difficult problem, too. Again, in order to correctly distinguish "`there were five to 10 million starving`" from "`he will speak at five to 10 million viewers`" we would have to do some "real" analysis, not just simple pattern matching. Therefore, we are trying to do only simple number standardization covering decimal points and thousand separators. Nevertheless, the recognition is language dependent:

2,8 → `<w num.orig>2,8 <f num.generated>2.8`
(Fr En It) 1'500 → `<w num.orig>1'500 <f num.generated>1500`

2.3 Separation of punctuation

The following punctuation receives separate tag `d` with empty text:

`-+<=>$#%&*@[]{}_|!"'(),./:;?^‘~ <TAB>`
e.g. ... found. → `...  <f>found <d period>`

2.4 Sentence boundaries

For any NLP system, it is important to know what forms a sentence. This problem is even harder than the previous two: many punctuation characters are ambiguous, the worst case being the dot ".". We adopted the view that it is better *not* to break a sentence in the middle; we would rather miss a

break. The rules rely on various lists, which classify some words according to their usage relative to the dot character.

The algorithm further supposes that the input text is "clean", i.e. no formatting, except for newline symbols.

This is a shortened description of the rules which govern the sentence boundary recognition:

```
Comments:

   <end-of-word> may be either <newline>
   or some delimiter other than one of the
   four from the "COUNTRY" specification file
   <C> means single capital letter
   <d ...>* means any sequence of delimiters
   <f (not)num> means anything not pure number

condition                                          break

? or ! or ; <newline>                              yes
ABBR. <newline>                                    no
IFNABBR. <newline> <f num>                         no
MARKABLE <end-of-word> <C>. <newline>              yes
NUMERABLE <end-of-word> <f num>. <newline>         yes
NONNUMABLE <end-of-word> <f num>. <newline>        no
<C>. <newline> <f cap>  (usu. names)               no
<f num>. <newline> MONTH                           no
<f num>. <newline> <f cap>                         yes
. <newline> <d ...>* <f cap>                       yes
<f (not)num>. <newline> <f num>                    yes
other                                              no
```

In the above rules, the words ABBR, IFNABBR, MARKABLE, NUMERABLE and NONNUMABLE are parameters representing sets of strings defined in language-dependent files.

2.5 Output SGML markup

The output markup has been chosen in such a way that it is simple to process even by programs with no specific SGML capabilities. Therefore, the tag names are one letter long. Each tag begins a separate line.

Conversion programs are provided to convert from entity-based ASCII (IRV, 7-bit) coding to ISO L1 and ISO L2 code table. For distribution purposes, we expect to use the former coding scheme until all the problems with accented characters are solved.

- Coding

  Two modes of accent coding:

  - Entity names, 7-bit coding ("ASCII")
  - ISO 8859-1 or ISO 8859-2 (8-bit coding)

- Markup

  - Document: `<bcpo>`
  - Paragraph: `<p id=...>`
  - Sentence: `<s id=...>`
  - Punctuation: `<d period>` `<d comma>` etc.
  - Punctuation separator: `<d nothing>`
  - Word Forms:
    `<f>` `<f cap>` `<f mixed>` `<f upper>` `<f abbr>`
    `<f cap.abbr>` etc.
  - Norm. Numbers: `<f num>` `<f num.generated>`
  - Original Numbers: `<w num.orig>`

Chapter 3

Morphology

3.1 Introduction

As it is quite well known these days, morphology has been deemed to be an uninteresting part of natural language processing, easily solvable by whatever means. This view, largely because of the simplicity of English morphology, has been changed when other languages came into the focus of computational linguistics. The turning point seems to be the publication of (Koskenniemi 1983), proposing the so called two-level model.

However, in the non-English speaking part of the world, morphology has been worked on many years before that. Slavonic languages in particular exhibit rich inflectional behavior, and that has been reflected in many publications. Czech is no exception, there have been many publications as well as real systems developed at different times for different purposes and on different computer platforms. Although they have explicitly described (large part of) Czech quite well, the required generality was usually missing. On the other hand, Koskenniemi's system (which had been inspired mainly by Karlsson, Karttunen, and Kay) was general enough to adequately handle many phonological and morphological phenomena, as was demonstrated later by many systems (for different languages) based on the two-level model.

However, the two-level model, as its name demonstrates, has been much better suited for general phonology than for general morphology. Obviously, it is possible to describe the morphotactics in the system (cf. the linked lexicon approach, actually a finite state network), but the modeling of long-

distance phenomena (which certainly do occur in morphology, cf. circumflexing in German *ge–t* (pass. part.) or Czech *nej–ější...* (superlative)) is awkward and unnatural.

The best solution would be to use the two-level model (or better, its phonology part) for phonology, and something else for morphology. However, the common sense says that morphology is "simple enough" to be handled by a unified system; it is thus this approach which has been chosen for the morphology formalism and software described in the present thesis.

3.2 The Basics: Input, Output and Between

The morphological component uses the results of the cleaning step as its input. This component analyzes each word-form unit, as identified by the markup step, individually and independently. The results are marked using SGML as well.

The morphology formalism is based on a context-free backbone with semantics expressed by means of feature structures and unification. This might seem to be a rather too strong tool for morphology, but it makes it relatively easy to use the same tool for different languages. Also, feature structures themselves are convenient (and popular) data representation format, which simplifies envisioned interfacing to parsers.

Currently, there are implementations of English, French, German and Czech, the latter using the most of the formalism's features.

An example of the desired output of:
... [bez] minulého parlamentu ...
(... [without the] last parliament ...)

```
...
<w>minul&eacute;ho
 <a> [hw=minul&yacute;,morf=[cat=[pos=a,a.type=ord],
        infl=[pf=[compar=base,nom.form=no,neg=no],
        raf=[gender=n|i,num=sg,case=gen]|[gender=a,
        num=sg,case=acc|gen]]]]
<w>parlamentu
 <a> [hw=parlament,morf=[cat=[pos=n],infl=[pf=[gender=i,
        num=sg,case=loc|dat|gen]]]]
...
```

3.3 Morphology - the Formalism

The basic idea of the described formalism is that words (in the sense defined earlier, see 2.1) are analyzed individually and independently. The formalism is capable of handling orthography/phonology, which enables us to use only one grammar formalism in our system taking care of the analysis all the way from surface to morphological categories.

3.3.1 Context-free skeleton

The basis of the morphology grammar consists of a context-free grammar skeleton. Left recursion is not allowed (due to the parsing style, see later).

The non-terminals are coded using angle brackets, terminal symbols are written as such, or quoted within doublequotes.

The CF rule format allows for easy specification of different types of morphotactic behavior, namely, suffixing, prefixing, circumfixing an infixing.

Skeleton examples:

```
<C> → <S> ed;
<R> → ge<Stem>t;
<Form> → im<AdjStem>;
<VForm> → <SepPrefix>ge<VStem1>en;
```

3.3.2 Semantics: Feature Structures

In the BCP morphology formalism, Feature Structures (Shieber 1986) can be seen as a special type of *attributes* in the sense of Attribute (CF) Grammars. Their basic function is to carry information from the bottom of the derivation tree up to the root and at the same time, to restrict the application of a rule of the grammar depending upon their value and conditions specified for such a rule.

The type of feature structures used is a slightly restricted form of the "standard" form widely used in grammars for natural language processing. The feature structures are not typed, and references within a single feature structure are not allowed. Negation is not allowed. Disjunction, however, is implemented in full. Obviously, references to feature values of **other** feature

structures are allowed (i.e., for referencing feature structures associated with right hand side nonterminals).

There is no "standard" or "automatic" unification imposed on the feature structures coming from the lower levels of the derivation tree. The value of the feature structure at the nonterminal node being built is composed using one or more rules, which explicitly specify the resulting feature structure value. Of course, unification, concatenation and references are the important tools to achieve the desired result. A rule is not applied, if the resulting feature structure is **nil**, i.e., when any of the values within such a feature structure is **nil**, which is the case when some unification fails.

3.3.3 Feature Structures: Coding

The feature structures are coded using the usual linear representation, with square brackets delimiting lists of feature-value pairs, commas separating individual feature-value pairs, and equal signs between the feature name and value. In references, feature names in a path to the referenced feature are separated by a carret sign (^), which is also used after the qualifying name of the relevant nonterminal. The "current" feature structure (associated with the left-hand side nonterminal) is denoted by a "star", "*". Disjunction of feature values is denoted by a vertical bar; parentheses are optional and do not have any effect on the evaluation. Literal strings (atoms) are written as such, except when they use characters with special meaning; then they must be quoted within doublequotes. Spaces, tabs and newlines do not matter. Case is significant.

Examples of feature structure values:

No disjunction, no references:

```
[a1=v1, a2=v2, a3 = [a31=v31]]
[a1=[b1="see$"],a2=NN0]
atom123
```

Disjunction:

```
a|b
[x=hd1|hd16|hd26]
[infl=[num=sg,case=(nom|acc)]|[num=pl,case=acc]]
```

References:

```
*^a1^x
[x=<S>^s, cat=<S>^pos, infl=[case=nom|acc]]
```

3.3.4 Concatenation of atoms

Two feature structure values which are atoms can be concatenated. The result is an atom. Concatenation of an atom and a feature structure proper or of two feature structures proper is **nil**, or fails.

The concatenation operation is denoted by a "+" sign.

Examples:

```
[a2=<D>+s, a3=<S>^s+y]
<_>+"s$"
```

3.3.5 Unification

The unification operation works as expected (Shieber 1986); two feature structures (values) unify iff all their features with the same name unify; atoms unify iff they are equal. An atom never unifies with a non-atomic feature structure.

If two feature structures unify, the result, again, is quite standard: for atoms, any of them (they are identical), for non-atomic feature structures, the union of all features which are present only in one of the feature structures with the results of the unification of the values of the remaining features (i.e., of those which are present in *both* feature structures).

The unification is denoted by the "#" operator.

Examples:

```
[a2=[<D>^x#hd1]]
<LEX>#stem
```

3.3.6 Building the LHS Feature Structure

For each rule, there is a sequence (possibly of length 1 only) of so called "assignments", which build the feature structure eventually associated with

the left hand side (LHS) nonterminal in the derivation tree being constructed.

Each of the assignments first creates a feature structure value, defined by the right hand side of the assignment. Then the resulting value is assigned to the feature specified at the assignment's left hand side, possibly to the nonterminal feature structure itself. In a following assignment, the value of the "current" feature structure can be used again when constructing the new value.

The qualifying "*" may be omitted as a qualifier on the left hand side of the assignment, including the case when the resulting feature structure value is assigned to the left hand side feature structure as a whole. Assignments are separated by commas, and the block of assignments associated with a single rule is terminated by a semicolon. The assignment symbol is, not surprisingly, an equal sign.

Examples:

```
 = [string=<S>^s, cat = v],
 analysis = *^x1^anal;
```

3.4 Morphology - the Dictionary

The dictionary can be seen as a Feature Structure Value, actually, a gigantic disjunction. Every "entry" forms a separate disjunction element. The smallest unit of the morphology formalism is a CF rule; therefore, the dictionary appears in the semantic part of a single rule with an empty right hand side:

```
<LEX> : =
     [stem=atom,hw=atom-atom,pos=noun,x=hd1] |
     [stem=nov,hw=nový-new,pos=adj,x=reg] |
     [stem=prac,hw=pracovat-to_work,pos=verb,x=ovatn] |
     ...
     → "";
```

Then, we can add such a rule to any rule which accesses the dictionary, and use its "upcoming" FS value in the semantic part where necessary (obviously, unification will play an important role in the constraints):

```
<R> → <S>$ <LEX> : =
     <LEX> # [x=hd1,stem=S,case=nom|acc,num=sg]
```

```
<R> → <S>u$ <LEX> : =
     <LEX> # [x=hd1,stem=S,case=gen|dat|loc,num=sg]
...
<R> → <S>ech$ <LEX> : =
     <LEX> # [x=hd1,stem=S,case=loc,num=pl]
<S> → <L> : = <L>;
<S> → <L><S> : = <L>+<S>;
<L> → a : = a;
<L> → A : = A;
<L> → b : = b;
<L> → B : = B;
...
<L> → Z : = Z;
```

The above (simplified, of course, as always) example shows how the stem string is described (nonterminal `S`, using `L` for description of all letters), and how the dictionary is "searched" for such a stem: the unification of `LEX` and a feature structure containing the `stem` attribute succeeds only if the stem is found somewhere in the dictionary. The symbol `$` denotes the end of input. The above example also shows how an additional constraint, namely, the `x` attribute, may be applied (see also later, Paradigms).

3.5 Handling various linguistic phenomena

3.5.1 Orthography - Capitalization

The simplest approach is to allow for capitalization anywhere in the input text:

```
<L> → a : = a;
<L> → A : = A;
<L> → A : = a;
<L> → b : = b;
<L> → B : = B;
<L> → B : = b;
...
<L> → z : = z;
```

```
<L> → Z : = Z;
<L> → Z : = z;
<S> → <L> : = <L>;
<S> → <L><S> : = <L>+<S>;
```

Note that this does not allow for input text decapitalization, i.e., it will not generate stem "George" if the input contains "george". However, should we want to do that, it is certainly easy to modify this grammar appropriately.

To keep the capitalization information, one could write:

```
<L> → a : = [l=a,cap=no];
<L> → A : = [l=A,cap=yes];
<L> → A : = [l=a,cap=yes];
<L> → b : = [l=b,cap=no];
<L> → B : = [l=B,cap=yes];
<L> → B : = [l=b,cap=yes];
...
<L> → z : = [l=z,cap=no];
<L> → Z : = [l=Z,cap=yes];
<L> → Z : = [l=z,cap=yes];
<S> → <L> : = <L>;
<S> → <L><R> : = [l=<L>^l+<R>,cap=<L>^cap];
<R> → <L> : = <L>^l;
<R> → <L><R> : = <L>^l+<R>;
```

and then use the feature `cap` on the upper levels of the derivation tree. This example handles capitalization at the beginning of a word; "all-uppercase" and mixed types would be just a little more complicated.

3.5.2 Phonology - "Umlaut"

```
<L> → a : = [l=a,umlaut=no];
     ...
<L> → ä : = [l=a,umlaut=yes];
<L> → ä : = [l=ä,umlaut=no];
     ...
<N> → <L> : = [l=<L>^l,umlaut=<L>^umlaut # no];
<N> → <L><N> : = [l=<L>^l+<N>^l,umlaut=<L>^umlaut # no];
```

```
<U> → <L> : = [l=<L>^l,umlaut=<L>^umlaut # yes];
<S> → <N> : = <N>;
<S> → <U><N> : = [l=<U>^l+<N>^l,umlaut=yes];
<S> → <N><U><N1> : = [l=<N>^l+<U>^l+<N1>^l,umlaut=yes];
<N1> → <N> : = <N>;
<R> → <S>e$ <LEX> : =
    <LEX> # [stem=<S>^l,umlautPL=<S>^umlaut,num=pl,...]
<LEX> → "" : = ...
    [stem=Baum,umlautPL=yes,...]  |
    [stem=Baumblast,umlautPL=no,...]  |
    ...;
```

Here, the <L>-rules simply introduce the features of the lexical characters, namely, whether the "umlaut" process took place when producing the surface text form or not. The nonterminal <N> accumulates strings of letters which were not subject to "umlauting". The nonterminal <U> always contains a single "umlauted" letter; it's l feature, however, already contains the lexical form of that letter. The unification of <L>^umlaut with either no or yes in these rules effectively disables their application in the unwanted cases. (Remember, if the value of any feature in the feature structure being constructed is nil, meaning the unification has failed, the associated rule cannot be applied.)

The nonterminal <S> (which is used to define the stem of a word form) then consist of non-"'umlauted" string of letters, or a string of letters which contains exactly one "umlauted" letter. The resulting feature structure at the <S> nonterminal then contains the umlaut feature, set accordingly.

Then it is easy to check, for particular categories (plural in this case), whether the value of the umlaut feature of the stem matches the lexical feature associated with that category; in this case, umlautPL.

[Certainly this example is oversimplified. Actually, if "ü" were handled in the same way, it overgenerates: the form "Baüme" would be incorrectly recognized as the plural form of "Baum". Additional measures must be taken to prevent this from happening.]

3.5.3 E-deletion

In this section, we present a slightly different approach to phonological processes than that shown for umlaut in the previous section. The stem change is indicated by a special "lexical" character, which then triggers the application of appropriate rules.

```
<L> → a : = [l=a];
<L> → b : = [l=b];
     ...
<N> → <L> : = [l=<L>^l];
<N> → <L><N> : = [l=<L>^l+<N>^l];
<S> → <N> : = <N>;
<S> → <N>e<L> : = [l=<N>^l+@E+<L>^l,Edeleted=no];
<S> → <N><L> : = [l=<N>^l+@E+<L>^l,Edeleted=yes];
<R> → <S> : =
     <LEX> # [stem=<S>^l,EdeletedTest=<S>^Edeleted # no,
              num=sg,case=nom|acc,...];
<R> → <S>e : =
     <LEX> # [stem=<S>^l,EdeletedTest=<S>^Edeleted # yes,
              num=sg,case=gen,...];
<LEX> → "" : = ...
     [stem=kop@Ec,...]  |
     [stem=vypínač,...]  |
     ...;
```

The nonterminal `<L>` describes individual letters, `<N>` a string of letters. The nonterminal `<S>` describes a word form, setting the feature `Edeleted` to indicate that e-deletion took place. This feature then constraints the application of rules assigning grammatical categories; in this case, e-deletion takes place only in genitive singular. The constraint `EdeletedTest` is irrelevant for words not containing "`@E`" at the lexical level, as the rule

```
<S> → <N> : = <N>;
```

(used in such cases) does not restrict the `Edeleted` value in any way.

3.5.4 Paradigms

```
<L> → a : = [l=a];
<L> → b : = [l=b];
     ...
<N> → <L> : = [l=<L>^l];
<N> → <L> <N> : = [l=<L>^l+<N>^l];
<S> → <N> : = <N>;
<R> → <S>uji : =
     <LEX> # [stem=<S>^l,x=ovatn,num=sg,pers=1,...];
     ...
<R> → <S> : =
     <LEX> # [stem=<S>^l,x=hd1,num=sg,case=nom|acc,...];
<R> → <S>u : = <LEX> # [stem=<S>^l,x=hd1,
     num=sg,case=gen,...];
<LEX> → "" : = ...
     [stem=hrad,x=hd1,...]  |
     [stem=kup,x=ovatn,...]  |
     ...
;
```

This is the simplest, but by far the most important use of unification for inflective languages such as Czech. In these languages, many categories are combined in a single ending, and at the same time, the endings are highly ambiguous. The worst "feature" of such morphological systems, however, is that for different words, the same ending can mean different things. Therefore, we need to have classes, or "patterns", or "paradigms", in order to avoid wrong word form interpretation. Actually, the same problem is known in English: the ending "s" is ambiguous in a well-known way. In English, however, the part of speech is sufficient to disambiguate its interpretation. In Czech, on the other hand, the same case appears within individual part-of-speech categories, even within more subtle classes of words: for example, the ending "`-i`" in the masculine animate noun class means either nominative plural (`dlouháni`, *the tall men*), or dative singular (`muži`, *[to a] man*). Obviously, these two words have to belong to different declension classes.

In the above example, it is shown how the paradigmatic constraints work. Each word in the dictionary is assigned an extra feature, the **paradigm**

name. Each rule of the grammar, which assigns the morphological category based on a recognized ending (in this case, the rules for nonterminal <R>), then contains the same feature with paradigm names for which the ending ↔ category correspondence is right. The unification mechanism then does the rest of the job.

3.6 Implementation

3.6.1 Dictionary

As has been already pointed out, the dictionary can be seen as a rule generating an empty string associated with a single "dictionary" feature structure. However, the simple approach to the unification operation would be extremely inefficient on such a structure; moreover, separating the dictionary is a good idea from the point of view of modifications, consistency, space, etc.

Therefore, instead of really writing the actual feature structure, we write each stem on a separate line in a source dictionary text file, distinguishing the stem string from the other features.

The following French dictionary feature structure segment

```
[hw=abolir,stem=abolir,x=v19,pos=v,vf=t]|
[hw=abdiquer,stem=abdiquer,x=v6,pos=v,vf=v]|
[hw=abat-jour,stem=abat-jour,x=nm,pos=nm]|
[hw=abats,stem=abats,x=nplm,pos=nm]|
[hw=avoir,stem=aurai,x=xv,mode=ind,tns=fut,pers=1,num=sg]|
[hw=avoir,stem=auras,x=xv,mode=ind,tns=fut,pers=2,num=sg]|
[hw=avoir,stem=aura,x=xv,mode=ind,tns=fut,pers=3,num=sg]...
```

is written as

```
abolir [x=v19,pos=v,vf=t]
abdiquer [x=v6,pos=v,vf=v]
abat-jour [x=nm,pos=nm]
abats [x=nplm,pos=nm]
aurai [hw=avoir,x=xv,mode=ind,tns=fut,pers=1,num=sg]
auras [hw=avoir,x=xv,mode=ind,tns=fut,pers=2,num=sg]
aura [hw=avoir,x=xv,mode=ind,tns=fut,pers=3,num=sg]
```

Note that if the hw feature value is the same as the stem value – which is often the case – the hw feature may be omitted. It is filled back automatically during subsequent processing.

This little change enables us to further simplify the dictionary coding; it is clear that feature structures like

```
[x=v6,pos=v,vf=v]
```

are going to be quite frequent. Others, however, like

```
[hw=avoir,x=xv,mode=ind,tns=fut,pers=2,num=sg] ,
```

are completely idiosyncratic. Thus, the dictionary is further divided into two parts: the so called "regular" dictionary (usually containing the most of the lexical items, like abolir, abdiquer, etc.), and an "irregular" dictionary, containing stems like aurai. The irregular part of the dictionary must be entered as shown above; the regular part is subject to preprocessing which basically expands feature structure abbreviations, or simple "macros" (called sometimes classes, or paradigms, as they often correspond to the individual inflectional paradigms), into a feature structure:

```
abolir vt
abdiquer vv
abat-jour nm
abats nmpl
```

Here, the class vt together with the infinitive ending information -ir causes an expansion of vt into

```
[x=v19,pos=v,vf=t] ;
```

similarly, nm is expanded into

```
[x=nm,pos=nm] .
```

3.6.2 Rules

Apart from technical details, such as various size specifications, the rules look much like what we have shown previously in the examples, except that all rules for a single nonterminal are grouped together; this allows to write only the right hand sides of rules:

```
%LHS;  R
{ regulier }
{ masculin en -s }
```

```
<_><C>$ : * = [stem=<_>^s+<C>^s,x=j,infl=[gen=m,num=sg]];
<_><J>$ : * = [stem=<_>^s+<J>^s,x=j,infl=[gen=m,num=sg]];
<_>e$ : * = [stem=<_>^s+e,x=j,infl=[num=sg]];
<_><C>s$ : * = [stem=<_>^s+<C>^s,x=j,infl=[gen=m,num=pl]];
<_><J>s$ : * = [stem=<_>^s+<J>^s,x=j,infl=[gen=m,num=pl]];
```

In this French example, the `<C>` and `<J>` nonterminals are defined elsewhere as consonants and vowels, respectively.

Also, it might seem that in this example, the actual dictionary unification is missing; not so, as for every left hand side nonterminal (i.e., for a group of rules having the same LHS), it is possible to define so called default actions, at the beginning of the sequence of assignments (see 3.3.6), after the first assignment, and at the end of the assignment sequence. In the above example, we actually have:

```
%LHS;  R
{ regulier }
{ masculin en -s }

%DEFAULT_ACTIONS;
%INIT; ;
%AFTER_1ST;
a = <LEX> # *;
%CLOSING;
* = *^a,
a=ANY,
stem=ANY,
x = ANY;

... [the rules come here]
```

The atom `ANY` represents unspecified value; at the final output formatting, such features are not printed. Thus specifying the assignment `feature-name = ANY` effectively deletes this feature from the resulting feature structure.

Typically, the constraining features are introduced at each rule as the first assignment, then the `AFTER_1ST` default applies, which contains the general

dictionary unification operation. The `CLOSING` assignments do some cleanup, then.

The feature which the dictionary is indexed on (`stem` in our examples) is also specified; it is the first feature name appearing in the list of all possible feature names, which must be specified in the source rules file. The name of the nonterminal representing the dictionary is also specified (`<LEX>` in our examples).

There is no need to put the dictionary nonterminal to every rule; it is present by default in all of them, enabling its free use in any assignment. However, there is one restriction put on the use of the dictionary nonterminal in unification: it cannot be used for referencing any features in the dictionary, it can be used only at the left hand side of the unification, and the feature structure with which it is being unified must contain the feature on which the dictionary is indexed.

3.6.3 Parsing

Parsing is done in the simplest possible way, i.e. top-down, depth-first. This puts one restriction on the grammar, namely, it must not contain left recursion.

The assignments are being evaluated at reduction time. The parser backtracks if any of the unifications involved fails.

There is a front-end to the core parser, which takes care of identifying words in the input text (which is not that difficult, as the input is assumed to be preprocessed by the cleaning & markup program discussed above, see 2), and of output formatting. Obviously, there are also tools for testing and debugging the morphological rules and the dictionary.

3.6.4 Implementation

The software (both markup and morphology, as well as other text processing tools developed at ISSCO) currently runs on Sun workstations (both Sun-3 and Sun-4), on a RS/6000 (AIX 3.2) and under Linux on the Intel 386 architecture. It should be fairly easy to port it to another machine running a similar operating system, as it have been all successfully compiled without problems using `gcc`.

At ISSCO, morphology grammars for English, German, French has been developed; the work on Czech has started at ISSCO and is currently being worked on at Charles University in Prague. The coverage of the grammars is quite substantial, with the exception of German they all contain more than 50,000 lexical units.

3.6.5 Availability

The BCP software containing the markup and morphology programs described herein can be obtained by writing to Susan Warwick-Armstrong at `susan@divsun.unige.ch` (ISSCO, Genève, Switzerland). It is available freely for research purposes; a written agreement must be signed, however. Quite substantial samples of morphology grammars for English, French, German and Czech are included; however, the Czech version is a little bit outdated now (and not even complete). The package includes also other programs developed at ISSCO (parallel text alignment program, and a parallel text viewer).

The Czech version is being completely rewritten as of now (May 1994) and will not be available until the end of 1994 at least. For details, write to the author.

Chapter 4

Czech Morphology

4.1 Output FS Definition

There are six main types of Feature Structures.[1] Here, the relevant general FS templates are presented (using attribute names only). Some of the attributes need not be used.

- nouns

```
[key,cat[pos],
     morf[infl[pf[gender,num,case]]]];
```

- adjectives

```
[cat[pos,a.type],
     morf[infl[pf[neg,compar,gender],
               raf[gender,num,case]]]]
```

- pronouns

```
[cat[pos,prn.type],
     morf[infl[pf[neg,compar,gender],
               raf[gender,num,case]]]]
```

[1] However, we do not really use *typed* feature structures.

- verbs

```
[cat[pos,v.type],
     morf[infl[pf[v.form,mode,tense,neg,compar],
               raf[pers,num]]]];
```

- adverbs

```
[cat[pos],morf[infl[pf[neg,compar]]]]
```

- other

```
[cat[pos]]
```

The `pos` feature has been (for some parts–of–speech values) subdivided into subclasses `a.type, prn.type, v.type`, according to its value. The `morf` feature contains the morphological categories, often using disjunction (for ambiguities). For the time being, it always contains only the `infl` feature for inflectional categories. The feature `pf` ("proper features") is used for inherently inner features of the word form. The feature `raf` ("requested agreement features") is used, on the other hand, for describing morphological categories used for surface agreement checking. The feature `neg` can take only two values: `yes` and `no`. The feature `compar` contains information about the comparative grade: `base` means not a comparative and not a superlative, `compar` comparative, `sup` superlative.

For verbs, the feature `v.form` indicates the verb form type (infinitive, imperative, fully inflected form). Other features are, by and large, self-explanatory.

4.2 The Inflectional System

In traditional grammars, *nouns, adjectives, pronouns, numerals and verbs* are considered to be the inflectional parts of speech (e.g., (Bělič 1981), (Havránek and Jedlička 1963), (Dokulil et al. 1986), (Petr 1986), (Šmilauer 1969) and (Daneš, Grepl and Hlavsa 1987)). Taking a closer look, however, one can see that

- pronouns

  are so wildly irregular, that it requires a separate paradigm for almost each of them (with the exception of those pronouns which morphologically behave as adjectives);

  on the other hand,

- adverbs

  form comparative and superlative in a fairly regular, at least most of them (in addition to being often derivatives of regular adjectives)

Therefore, five groups of paradigms have been established for our purposes:

1. nouns
2. adjectives
3. adverbs
4. verbs
5. uninflected

It should be noted that these groups reflect the morphological behavior of words, not their actual (syntactic) part of speech. The part-of-speech information is assigned using the associated feature structures, either based on the paradigm (certainly many paradigms, especially for nouns and verbs, are unambiguously pointing to the correct part-of-speech), or based on the information provided explicitly in the dictionary entry.

Having the possibility to assign part-of-speech and other information in the dictionary entries, the *uninflected* group consists of a single paradigm, namely, `0`, meaning "do not do anything to the root (when producing the surface form)".

There is no "irregular" paradigm group; really irregular words are entered into the irregular part of the dictionary inflected and assigned class `0` (the "do-not-do-anything" class of the *uninflected* group). This is the case, e.g., for most pronouns.

In order to achieve compatibility with other software systems which use some form of morphological processing, the phonological changes have been made part of the paradigm. Thus, it was necessary to have much finer subclassification of paradigms than the traditional grammars would suggest. For example, the masculine inanimate nouns with so called "hard consonant" in nominative singular (normally considered to be the finest paradigm class) were split into 16 subgroups, depending on the genitive singular, locative singular and locative plural variation as well as on the stem changes and/or e-deletion. (For a detailed discussion of nouns, see Section 5.1.) Similar refinements have been made for other parts of speech (see below).

Derivations has been made part of the respective paradigms when they appear to be regular. For nouns, this concerns possessive adjectives derived from proper names or nouns denoting persons (*-ův, -in*). For adjectives, the derived adverb (*-ý/-í* → *-ě* with variations depending upon the stem-final consonant), *-cký/-ský* → *-cky/-sky*, and usually also the property noun (*-ost* are formed regularly. Most numerals form the "-times" *-krát(e)* derivative, as well as the "fraction" derivative (*-ina*). Verbs are most productive forming derivatives: counting iterative forms, as much as 28 derivatives can be produced from a single stem.

The inclusion of derived forms into the basic paradigms influences most the coverage of the dictionary. Therefore, while some of the dictionary entries represent irregular forms and for a single lexical unit there are up to 15 entries, some entries represent many lexical units, or traditional "words". It is thus misleading to count the number of dictionary entries to find out the number of words covered by the dictionary; the Czech dictionary used in the presented system here has over 110 thousand entries representing probably almost twice as much traditional "words". However, some of the derived words are potential, at best (but not wrong) – the only criterion for sufficient coverage is a field test, of course. [2]

[2] As it turns out, in some applications (e.g., information retrieval) many derived forms can have different meaning and thus, they should not be merged with the base verb (adjectives are affected, too). Therefore, in some of such applications, it is necessary to remove all derivatives from the base paradigms, add entries to the dictionary, and use something like "derivation thesaurus" in cases when the derivation does *not* change meaning. Actually, negation must be sometimes handled in the same way: certainly "nezaměstnaný" (unemployed) should not be mapped to "zaměstnat" (to employ), or not even to "zaměstnaný" (employed)).

- group: nouns

  The paradigms in this groups follow, by and large, the traditional scheme of Czech noun classification: there are 14 basic classes, plus two more for adjective-like declension:

  1. masculine animate, hard consonant sg. nom. ending
  2. masculine animate, soft consonant sg. nom. ending
  3. masculine inanimate, hard consonant sg. nom. ending
  4. masculine inanimate, soft consonant sg. nom. ending
  5. feminine, sg. nom. ending -*a*
  6. feminine, sg. nom. ending -*e*
  7. feminine, sg. nom. ending -*ost*
  8. feminine, sg. nom. consonant ending (not -*ost*)
  9. neuter, sg. nom. ending -*o*
  10. neuter, sg. nom. ending -*e*, young animals (and some others)
  11. neuter, sg. nom. ending -*e*, not young animals
  12. neuter, sg. nom. ending -*í*
  13. masculine animate, sg. nom. ending -*a*
  14. masculine animate, sg. nom. ending -*e*
  15. masculine animate, sg. nom. ending -*ý*
  16. feminine, sg. nom. ending -*á*

  These classes were further divided into finer *elementary classes*, according to form variations, stem changes, e-deletion, etc. There are also some extra classes for proper names, derived possessive adjectives, and words of foreign origin. For the complete list of noun classes, see Appendix A and Section 5.1.

- group: adjectives

  Czech adjectives are simple in principle. Traditionally, they are divided into two subgroups: so called "hard" (masculine animate ending

in nominative singular: *-ý*), and "soft" (nominative singular *-í*). However, there are some stem changes (*-ský, -cký, -rý, -chý*) taking place (always in nominative plural), and some adjectives cannot form negative (e.g. those already in negative form) and/or the corresponding adverb (or the adverb is formed irregularly); therefore, there are 8 adjective classes. One special class (`0n`) allowing only negation (but no inflection) is added in order to reduce the number of irregular forms in the dictionary by half (as they usually form negation). The description of negation/degrees of comparison rules follows in Section 4.3.4. For the complete list of adjective classes, see Appendix C (first page).

- group: adverbs

  Degrees of comparison and negation is the only "inflection" possible for adverbs. Nevertheless, there are two classes designed specifically for adverbs: one (`jev`) for those ending by *-ě* and `ev` for those ending by *-e*. The description of negation/degrees of comparison rules follows in Section 4.3.4. For the complete list of adverb classes, see Appendix C (first page).

- group: verbs

  There are traditionally five "superclasses" of Czech verbs based on the 3rd person singular present tense ending:

  1. *-e*
  2. *-ne*
  3. *-je*
  4. *-í*
  5. *-á*

  Each of them is traditionally further divided into classes:

  1. *-e*
     (a) *bere (brát)*
     (b) *nese (nést)*
     (c) *peče (péct)*

   (d) *maže (mazat)*
   (e) *tře (třít)*
2. *-ne*
   (a) *tiskne (tisknout)*
   (b) *mine (minout)*
   (c) *začne (začít)*
3. *-je*
   (a) *kryje (krýt)*
   (b) *kupuje (kupovat)*
4. *-í*
   (a) *prosí (prosit)*
   (b) *trpí (trpět)*
   (c) *sází (sázet)*
5. *-á*
   (a) *dělá (dělat)*

giving 14 different conjugation classes.

The morphology described here basically follows this classification scheme, but the classes had to be refined and some "specific treatment" had to be introduced to handle Czech conjugation in its entirety.

As it turns out (not surprisingly), the regularity proportional to the number of words belonging to a given class. Given the usual trade-off between "purity" and effectiveness, it was necessary to treat these classes differently.

The largest classes, represented by the verbs *tisknout*, *kupovat* and *dělat* are completely regular, including derivations. The naming convention tries to be mnemotechnical, namely, according to the infinitive ending of the group. The identification of the aspect is appended to the end of the identifier (`n` stands for imperfective (*nedokonavé*), `d` for perfective (*dokonavé*)):

```
kup ovatn =kupovat
děl atn =dělat
vytisk noutd =vytisknout
```

The class represented by *prosit* is regular, although the verbs ending by *-dit, -tit* and *-nit* require special treatment because of stem changes in imperative and there are some differences in passive forms, too. There are also some verbs in this class which are more irregular than others (e.g. *prosit* itself, despite being traditionally chosen as the class representative). This has led to so called partial classes (already used for some nouns and adjectives), which do not contain all the endings for the lexical unit; irregular endings are excluded. Much less entries is then needed over listing of *all* forms:

```
pros itxn =prosit # all forms except the following:
pros i1 =prosit # imperative
pros t3n =prosit # transgressive
prošen s2 =prosit # passive & some derivations
```

All of the "regular" classes are nevertheless divided into two paradigms, one for imperfective aspect verbs and one for perfective aspect. The difference between them is the set allowed transgressive forms.

For the other classes, the concept of a partial class has been developed even further. It appears that the irregularity of many verbs (irregularity means here that very few verbs use the same conjugation pattern) is demonstrated in the stem alternations, not in the endings. Moreover, the partial classes correspond to morphological categories: infinitive, present tense, past tense, passive participle, transgressive, imperative, derivations. Using this scheme, at most seven stems (i.e. dictionary entries) are needed for a single lexical unit. For example, the verb *bát* (to be afraid) has been split into six entries:

```
bá t =bát # infinitive
boj i1 =bát # imperative
boj p3 =bát # present tense
boj t3n =bát # transgressive (present)
bál r =bát # past participle
bán ns =bát # passive & derivations
```

Of course, there are really irregular verbs: *být*, *mít*, and *moci*. But still, even for these some of the partial classes can be used (example for *být*, "to be"):

```
buď i1 =být
bý t =být
bud p1 =být
byv md =být
byt ns =být
jsem 0ns =být
jsi 0ns =být
je 0 =být
jest 0 =být
není 0 =být
jsme 0ns =být
jste 0ns =být
jsou 0ns =být
js t1n =být
byl r =být
bych 0 =být
bys 0 =být
by 0 =být
bychom 0 =být
byste 0 =být
abych 0 =být
abys 0 =být
aby 0 =být
abychom 0 =být
abyste 0 =být
kdybych 0 =být
kdybys 0 =být
kdyby 0 =být
kdybychom 0 =být
kdybyste 0 =být
```

Concerning the often discussed problem of prefixes, it is believed that it is impossible to derive the meaning of a prefixed verb from the base lexical unit in a compositional manner. Therefore there is no attempt to do that. On the other hand, the need for accessing the appropriate aspect pairs (imperfective/perfective, and iterative as well) is considered valid, but it is not implemented in the present system (with the

exception of iteratives).

- group: uninflected

  From the morphological point of view, all of the uninflected words belong to a single class, 0. See also discussion of irregularities in Sections 4.3.3 and 4.2.

4.3 Rules

4.3.1 Predefined Attributes, Start-symbol and Dictionary Nonterminal

```
%ATTRIBUTES;
key,
hw,
x,
cat,
pos,
a.type,
lpart,
rpart,
seq,
an,
morf,
infl,
prn.type,
prn.subtype,
v.type,
conj.type,
pf,
raf,
gender,
num,
case,
compar,
nom.form,
```

```
neg,
v.form,
pers,
mode,
tense,
aspect,
rtense
;

%START;
START

%DICTIONARY_ROOT;
LEX
```

4.3.2 Capitalization and Numbers

At the lowest abstract level, the rules handle orthographical conventions, such as capitalization; all 41 Czech letters are therefore described using nonterminals, and other nonterminals are introduced for the description of various possibilities of capitalization.

In addition of having special nonterminals for all lowercase, uppercase, and uppercased letters, there are also nonterminals for each letter, be it lowercase or uppercase; these are being used in the rules to specify prefixes and suffixes, where the case does not matter.

```
%LHS;  LC
%RHS;
a :  = a;
b :  = b;
c :  = c;
...

%LHS;  UC
%RHS;
A :  = A;
B :  = B;
```

```
C :  = C;
...

%LHS;  ULC
%RHS;
A :  = a;
B :  = b;
C :  = c;
...

%LHS; a
%RHS;
A :  = a;
a :  = a;

%LHS; b
%RHS;
B :  = b;
b :  = b;

%LHS; c
%RHS;
C :  = c;
c :  = c;

...

%LHS;  LCC
%RHS;
<LC> : = <LC>;

%LHS;  ALC
%RHS;
<LCC> : = <LCC>;
<ULC> : = <ULC>;

%LHS;  ANY_CHAR
```

```
%RHS;
<LCC> : = <LCC>;
<UC> : = <UC>;

%LHS;  ALW
%RHS;
<ALC> : = <ALC>;
<ALC><ALW> : = <ALC><ALW>;

%LHS;  LW
{ lower-case word }
%RHS;
<LCC> : = <LCC>;
<LCC><LW> : = <LCC><LW>;

%LHS;  ANY_STR
%RHS;
<ANY_CHAR> : = <ANY_CHAR>;
<ANY_CHAR><ANY_STR> : = <ANY_CHAR><ANY_STR>;

%LHS;  ULW
{ word containing upper-case letter - lowered }
%RHS;
<ULC> : = <ULC>;
<ULC><ALW> : = <ULC><ALW>;
<LW><ULC> : = <LW><ULC>;
<LW><ULC><ALW> : = <LW><ULC><ALW>;

%LHS;  UW
{ word beginning with upper-case letter }
%RHS;
<UC> : = <UC>;
<ULC> : = <ULC>;
<UC><LW> : = <UC><LW>;
<UC><ULW> : = <UC><ULW>;
<ULC><LW> : = <ULC><LW>;
<ULC><ULW> : = <ULC><ULW>;
```

```
%LHS;  _
{ any string }
%RHS;
<LW> : = <LW>;
<UW> : = <UW>;
```

Numbers are handled in a simple way, as some semi-intelligent preprocessing is assumed (see 2.2):

```
%LHS;  NUM
%RHS;
1 := 1;
2 := 2;
3 := 3;
4 := 4;
5 := 5;
6 := 6;
7 := 7;
8 := 8;
9 := 9;
0 := 0;
. := .;

%LHS; NUMBER
%RHS;
<NUM> : = <NUM>;
<NUM><NUMBER> : = <NUM><NUMBER>;
```

4.3.3 Main Block of Rules

This block handles the core of the Czech morphological analysis, namely, it handles all declension and conjugation classes. Regular negation, degrees of comparison and numbers are handled separately.

```
%LHS; R
{ main block of rules }
```

```
%DEFAULT_ACTIONS;
%INIT; ;
%AFTER_1ST;
an = <LEX> # *;
%CLOSING;
* = *^an,
key = ANY,
an = ANY;

%RHS;
```

Irregular Forms

There must be a rule for accepting irregular forms, although no new information is being added:

```
<_>$ : = [key=<_>, x=0];
```

Similar rules are used for the classes `0n` (adjectives with negation) and `0ns` (verbs with negation), except that negation is allowed.

Regular or Partial Forms

For the discussion of the Czech inflectional system, see Section 4.2.

The application of the main block of "regular" rules is straightforward; the key to success is the attribute `x`, which takes care of properly filtering out impossible combinations of a root and an ending.

Some examples:

Dative/locative case singular of animate masculine nouns (declension classes `mz5`, `mz5i` or `mz5x`):

```
<_><c><o><v><i>$ : = [key=<_>ec, x=(mz5|mz5i|mz5x), cat=[pos=n],
     morf=[infl=[pf=[gender=a, num=sg, case=(dat|loc)]]]];
```

Adjectives, "soft" declension (without regard to negation and the derived adverb), all possibilities of interpretation for the base form (*-í* ending):

```
<_><í>$ :  = [key=<_>í, x=(i|ih|iv), cat=[pos=a, a.type=ord],
    morf=[infl=[pf=[compar=base, nom.form=no],
```

```
     raf=([gender=a, num=sg, case=(nom|voc)]
      | [gender=(i|n), num=sg, case=(nom|acc)]
      | [gender=f, num=sg]
      | [num=pl, case=(nom|acc|voc)])]]];
```

Possessive adjective derived from a feminine noun, which is itself a counterpart to a masculine noun; one possible class (`pn6fy`) only:

```
<_><r><y><n><i><n><u>$ : = [key=<_>r,x=pn6fy,
     cat=[pos=a,a.type=poss],
     morf=[infl=[pf=[gender=f],
                 raf=([gender=(a|i|n),num=sg,case=(dat|loc)]
                    | [gender=f,num=sg,case=acc])]]];
```

3rd person plural, present tense, verb class represented by *tisknout*:

```
<_><n><o><u>$ : = [key=<_>nout,x=(noutn|noutd),
     cat=[pos=v,v.type=full],
     morf=[infl=[pf=[v.form=fin,mode=ind,
                     tense=pres],
                 raf=[pers=3,num=pl]]]];
```

Example listings of some paradigm classes are presented in the Appendix D.

4.3.4 Negation, Degrees of Comparison, Numbers

As numbers are considered to be word forms, they are recognized, too:

```
%LHS; START
%RHS;
<WF> := <WF>;
<NUMBER>$ : an = [hw=<NUMBER>, pos=numid];
```

NB: These two rules are also the only two rules for the starting symbol of the Czech morphology grammar, see also Section 4.3.1.

`WF` represents a word form proper; degrees of comparison and negation (the only prefix operations recognized) are handled in a straightforward way:

```
%LHS; NEG
%RHS;
<R> := <R> # [cat=[pos=(n|prn|num|prep|conj|intj|prtcl|0)]];
<R> := <R> # [cat=[pos=a,a.type=poss]];
<R> := <R> # [x=(y|ye|cky|sky|yx|i|ih|iv),cat=[pos=a,a.type=ord],
              morf=[infl=[pf=[neg=no]]]];
<R> := <R> # [x=(inx|ynx),cat=[pos=a,a.type=ord]];
<R> := <R> # [cat=[pos=(v|ad)],morf=[infl=[pf=[neg=no]]]];
<n><e><R> :=
       <R> # [cat=[pos=a,a.type=ord],morf=[infl=[pf=[neg=yes]]]];
<n><e><R> :=
       <R> # [cat=[pos=(v|ad)],morf=[infl=[pf=[neg=yes]]]];
<n><e><R> :=
       <R> # [x=(y|ye|cky|sky|yx|s2|s4|s5|ns|a|ae|noutn|ovatn|
              it0n|itin|iten|titn|ditn|nitn|etn|jetn|atn|noutd|
              ovatd|it0d|itid|ited|titd|ditd|nitd|etd|jetd|atd),
              cat=[pos=n]], hw=ne+*^hw;


%LHS; DEG
%RHS;
<NEG> := <NEG> #
          [cat=[pos=(n|prn|num|prep|conj|intj|prtcl|0)]];
<NEG> := <NEG> #
          [x=(t1n|t2n|t3n|t4n|i1|i2|i3|p1|p2|p3|p4|p5|p6|p7|r|
              re|rx|s1|s2|s3|s4|s5|t|noutn|noutd),cat=[pos=v]],
              morf:infl:pf:aspect=*^aspect,aspect=ANY;
<NEG> := <NEG> #
          [x=(md|ovatn|it0n|itin|iten|titn|ditn|nitn|itxn|etn|
              jetn|atn|ovatd|it0d|itid|ited|titd|ditd|nitd|
              itxd|etd|jetd|ard),cat=[pos=v]];
<NEG> := <NEG> # [cat=[pos=a,a.type=poss]];
<NEG> := <NEG> # [x=(inx|ynx|yx|i|ih),cat=[pos=a,a.type=ord]];
<NEG> := <NEG> # [x=(y|ye|cky|sky|iv),cat=[pos=a,a.type=ord],
              morf=[infl=[pf=[compar=(base|comp)]]]];
<NEG> := <NEG> #
          [cat=[pos=ad],morf=[infl=[pf=[compar=(base|comp)]]]];
<n><e><j><NEG> :=
         <NEG> # [cat=[pos=a,a.type=ord],
```

```
              morf=[infl=[pf=[compar=sup]]]];
<n><e><j><NEG> :=
         <NEG> # [cat=[pos=ad],morf=[infl=[pf=[compar=sup]]]];

%LHS; WF
%RHS;
<DEG> := <DEG>, morf:cat=*^cat, cat=ANY, x=ANY;
```

The long lists of restrictions make sure that words belonging to paradigms not forming negation and/or degrees of comparison are *not* recognized.

4.4 Dictionary

4.4.1 Regular Part

The regular part of the source dictionary has the form (see 3.6.1)

```
<root> <pattern> =<lexical-form>
```

An example:

```
pomeranč sj1 =pomeranč
pomerančov y =pomerančový
pomerančovní hd1k =pomerančovník
pomet atn =pometat
pomet mt8e =pometlo
pometlář mz1z =pometlář
pometlář sky =pometlářský
pomezn i =pomezní
pomezí st =pomezí
pomešk atd =pomeškat
pomil ovatd =pomilovat
pomin p1 =pominout
pominou t =pominout
pominul r =pominout
pominut s2 =pominout
pominuv md =pominout
pomis i1 =pomísit
```

```
pomiň i1 =pominout
poml zn6 =pomlka
pomlask atd =pomlaskat
pomlask noutd =pomlasknout
pomlask ovatn =pomlaskovat
pomlať i1 =pomlátit
pomlel r =pomlít
pomlet s2 =pomlít
pomlev md =pomlít
pomlkov y =pomlkový
pomlouv atn =pomlouvat
pomlouvač mz1z =pomlouvač
pomlouvačn y =pomlouvačný
pomluv it0d =pomluvit
pomluv zn1 =pomluva
pomluviteln y =pomluvit
pomlácen s2 =pomlátit
pomlát itxd =pomlátit
pomlátiteln y =pomlátit
pomláz zn6e =pomlázka
pomlázkov y =pomlázkový
pomlč i1 =pomlčet
pomlč p3 =pomlčet
pomlč zn6e =pomlčka
pomlče s4 =pomlčet
pomlčel r =pomlčet
pomlčev md =pomlčet
pomlčkov y =pomlčkový
pomlí t =pomlít
pomn i3 =pomenout
pomnož ited =pomnožit
pomnožiteln y =pomnožit
pomnožn y =pomnožný
pomněn zn6e =pomněnka
pomněneč zn6e =pomněnečka
pomněnkov y =pomněnkový
pomní hd1k =pomník
```

```
pomníkov y =pomníkový
pomníkář mz1z =pomníkář
pomníkář sky =pomníkářský
pomníč hd1ek =pomníček
pomoc ps18 =pomoc
pomoci 0ns =pomoci
pomocn pn16ik =pomocník
pomocn y =pomocný
pomocni cky =pomocnický
pomoct 0ns =pomoci
pomodl i2 =pomodlit
pomodl itxd =pomodlit
pomodlen s2 =pomodlit
pomodliteln y =pomodlit
pomodř i2 =pomodřit
pomodř itxd =pomodřit
pomodřen s2 =pomodřit
pomodřiteln y =pomodřit
pomoh md =pomoci
pomohl r =pomoci
pomohou 0ns =pomoci
pomohu 0ns =pomoci
pomokř i2 =pomokřit
pomokř itxd =pomokřit
pomokřen s2 =pomokřit
pomokřiteln y =pomokřit
pomolo pn19 =pomolog
pomologi cky =pomologický
pomologi ns1 =pomologie
```

For sets of possible endings for a given `<pattern>`, see Appendices A and C.

After dictionary preprocessing, this dictionary fragment becomes

```
pomeranč [x=sj1]
pomerančovník [x=hd1k]
pomerančový [x=y]
pometat [x=atn]
```

pometlo [x=mt8e]
pometlář [x=mz1z]
pometlářský [x=sky]
pomezní [x=i]
pomezí [x=st]
pomeškat [x=atd]
pomilovat [x=ovatd]
pominout [x=t,hw=pominout]
pominu [x=p1,hw=pominout]
pominul [x=r,hw=pominout]
pominut [x=s2,hw=pominout]
pominuv [x=md,hw=pominout]
pomis [x=i1,hw=pomísit]
pomiň [x=i1,hw=pominout]
pomlaskat [x=atd]
pomlasknout [x=noutd]
pomlaskovat [x=ovatn]
pomlať [x=i1,hw=pomlátit]
pomlel [x=r,hw=pomlít]
pomlet [x=s2,hw=pomlít]
pomlev [x=md,hw=pomlít]
pomlka [x=zn6,hw=pomlka]
pomlkový [x=y]
pomlouvat [x=atn]
pomlouvač [x=mz1z]
pomlouvačný [x=y]
pomluva [x=zn1]
pomluvit [x=it0d]
pomluvitelný [x=y,hw=pomluvit]
pomlácen [x=s2,hw=pomlátit]
pomlátit [x=itxd,hw=pomlátit]
pomlátitelný [x=y,hw=pomlátit]
pomlázka [x=zn6e]
pomlázkový [x=y]
pomlč [x=i1,hw=pomlčet]
pomlčel [x=r,hw=pomlčet]
pomlčet [x=s4,hw=pomlčet]

pomlčev [x=md,hw=pomlčet]
pomlčka [x=zn6e]
pomlčkový [x=y,hw=pomlčkový]
pomlčím [x=p3,hw=pomlčet]
pomlít [x=t,hw=pomlít]
pomni [x=i3,hw=pomenout]
pomnožit [x=ited]
pomnožitelný [x=y,hw=pomnožit]
pomnožný [x=y]
pomněnečka [x=zn6e]
pomněnka [x=zn6e]
pomněnkový [x=y]
pomník [x=hd1k]
pomníkový [x=y]
pomníkář [x=mz1z]
pomníkářský [x=sky]
pomníček [x=hd1ek]
pomoc [x=ps18]
pomoci [x=0ns,hw=pomoci]
pomocnický [x=cky,hw=pomocnický]
pomocník [x=pn16ik,hw=pomocník]
pomocný [x=y,hw=pomocný]
pomoct [x=0ns,hw=pomoci]
pomodlen [x=s2,hw=pomodlit]
pomodli [x=i2,hw=pomodlit]
pomodlit [x=itxd,hw=pomodlit]
pomodlitelný [x=y,hw=pomodlit]
pomodřen [x=s2,hw=pomodřit]
pomodři [x=i2,hw=pomodřit]
pomodřit [x=itxd,hw=pomodřit]
pomodřitelný [x=y,hw=pomodřit]
pomoh [x=md,hw=pomoci]
pomohl [x=r,hw=pomoci]
pomohou [x=0ns,hw=pomoci]
pomohu [x=0ns,hw=pomoci]
pomokřen [x=s2,hw=pomokřit]
pomokři [x=i2,hw=pomokřit]

```
pomokřit [x=itxd,hw=pomokřit]
pomokřitelný [x=y,hw=pomokřit]
pomolog [x=pn19]
pomologický [x=cky]
pomologie [x=ns1]
```

actually meaning (see 3.6.1; in the Czech system, the dictionary is indexed using the attribute `key`):

```
 ...  |
[key=pomeranč,x=sj1]|
[key=pomerančovník,x=hd1k]|
[key=pomerančový,x=y]|
[key=pometat,x=atn]|
[key=pometlo,x=mt8e]|
[key=pometlář,x=mz1z]|
[key=pometlářský,x=sky]|
[key=pomezní,x=i]|
[key=pomezí,x=st]|
[key=pomeškat,x=atd]|
[key=pomilovat,x=ovatd]|
[key=pominout,x=t,hw=pominout]|
[key=pominu,x=p1,hw=pominout]|
[key=pominul,x=r,hw=pominout]|
[key=pominut,x=s2,hw=pominout]|
[key=pominuv,x=md,hw=pominout]|
...  etc.
```

4.4.2 Irregular Part

The irregular forms are simply listed here in the basic format (see 3.6.1), using the pattern `0` (zero) which means that only identical word forms match these entries.

An example:

```
je [hw=být,x=0,cat=[pos=v],
     morf=[infl=[pf=[v.form=fin,mode=ind,tense=pres],
         raf=[pers=3,num=sg]]]]
```

```
je [hw=oni,x=0,cat=[pos=prn],
     morf=[infl=[pf=[prn.type=pers,gender=a,num=pl,case=acc]]]]
kde [hw=kde,x=0,cat=[pos=adv],
     morf=[infl=[pf=[adv.type=q]]]]
komu [hw=kdo,x=0,cat=[pos=prn],
     morf=[infl=[pf=[prn.type=q,gender=(a|f|n),num=sg,case=dat]]]]
více [hw=hodně,x=0,cat=[pos=adv],
     morf=[infl=[pf=[compar=comp,neg=no]]]]
a [hw=a,x=0,cat=[pos=conj]]
     morf=[infl=[pf=[conj.type=c]]]]  ...  etc.
```

Chapter 5

Dictionary Tools

5.1 Semiautomatic Classification of Nouns (batch style)

5.1.1 Introduction

In many NL-processing tasks, the need for some kind of a dictionary is so strong that it is impossible to avoid it. When creating such a dictionary for inflectional languages, the information about the morphological behavior of words is one of the basic items in the dictionary entry. If there is a specialized, machine-readable dictionary, prepared by a linguist, big enough, no problems arise. Unfortunately, this is usually not the case. Moreover, no dictionary is ever complete - so new entries have to be added continuously.

Equipping the lexical entries with morphological information is an unpleasant task; very boring for linguists, and error-prone for anybody. And if the dictionary is to be updated primarily by non-linguists, the need for (at least some) automation is obvious.

Fortunately, some inflectional languages (including Czech, as well as the other Slavonic languages) tend to indicate their morphological properties by (some of) the written forms of the word itself, at least statistically. We tried to combine all accessible knowledge on Czech nouns declension into tables, which (when built into programs) could drive the (semi-)automatic process of insertion of morphological properties into dictionary entries.

For this purpose, we have chosen a very simple approach to the description

of Czech declension, excluding any intermediate levels between phonetics (or graphics) and morphotactics. It means that

- we deal with the morphotactics directly on the written text level;
- the structure of the morphological system will not be in accordance with the tradition.

We decided to employ the paradigm-based model; from this, and from the two basic presumptions given above it follows that

1. the number of paradigms will be rather large (for Czech or other Slavonic languages, of course)
2. there will exist (non-productive) classes for individual, highly irregular nouns.

As the purpose of this work is to facilitate morphological classification of new words being added to a dictionary, and new words are mostly regular, we can suppose that the irregular words which would belong to the class mentioned under (2) above are already in the dictionary. Further, we suppose that for nouns, all the morphotactics can be expressed by the rule

```
<form> -> <stem><ending>
```

both for nouns as well as their nominal derivatives.

So, the paradigms have the form of a set of endings (possibly with the morphological categories belonging to particular endings). The morphological information attached to each noun thus consists of two parts:

- the stem (or an information how to derive the stem from the basic dictionary form, if only the latter is present in the dictionary),

  and

- the paradigm name (which points to the appropriate set of endings).

5.1.2 The System of Czech Declension

In Czech, nouns are traditionally divided into four classes according to gender: masculine animate, masculine inanimate, feminine and neuter. For the purpose of this work, mainly because of possible word derivations, we divided the feminine category into two: feminine animate and feminine inanimate. No such division is of any help for neuters.

The division into classes according to genders helps to algorithmically assign the appropriate paradigm, and, from the point of view of our goal, is well-known also for non- linguists (because it is taught already at the elementary school level and at the same time it is simple enough and close to everybody's intuition to remember).

The deeper traditional division of the gender classes into subclasses according to the stem ending (soft, hard, vowel, etc.), although also taught in elementary schools, has proved not to be suitable for the automatic classification, for four reasons:

1. nobody remembers;
2. it is often too rough (variations, derivations and/or fusions among classes exist);
3. it is often unnecessary to enter such information, because it is contained in the shape of the ending of the dictionary form (nominative singular) of the noun;
4. it supposes some phonological (= orthographical) processing, which we have not included into this particular model.

The fourth reason is not too significant, because it only means that the number of paradigms is rather large (because they must reflect stem/ending changes which would be normally performed at the lower level), but from the classification point of view, this does not complicate matters nor makes them easier.

The system works for all syntactic nouns, i.e., for all words which in the usage play the role of a noun, with the exception of pronouns (which are a closed group and thus are supposed to be all included in any dictionary). Further, it works also for other parts of speech which morphologically behave like a noun according to the scheme <form> → <stem><ending>. In our

case, this is true for possessive adjectives, and for proper adjectives which do not form negation nor gradation. Moreover, we suppose that highly irregular nouns (e.g., those with irregular shortening of the stem vowel, or those with two or more very different stems) are already in the dictionary, so in most cases we do not cover them in the system.

5.1.3 The Division into Classes

For many regular nouns (the term "noun" is used here and henceforth in the sense of all word types covered by the system, in the sense of the last paragraph of the previous section), the information about their gender is sufficient enough to recognize (according to their ending in nominative singular form) to which paradigm they belong. This is especially true for feminine inanimate (class `f`), feminine animate (class `fn`) and most neuters (class `n`; the remaining neuters - those with adjective-like declension - have the class `nadj`). A more complicated situation arises for masculines, especially for the masculine inanimate; unfortunately, these form the majority of nouns when expanding dictionaries.

For masculine animate, in some cases, the ending of genitive singular is necessary to classify the noun properly; the nominative singular form is not enough. For example, the ending *-z* is ambiguous: *plaz* (the reptile) belongs to the paradigm `pn1` *(Gsg plaza, Dsg plazu/plazovi, ...)*, but *kněz* (the priest) belongs to `mzs1` *(Gsg kněze, Dsg kněži, ...)*; similarly for *-l*: *kutil* (the handiman) has *Gsg kutila, Dsg kutilu/kutilovi*, but *král* (the king) has *Gsg krále, Dsg králi*, etc.). Although for most masculine animate nouns the nominative singular ending is characteristic for a particular paradigm, we keep three basic classes for them, based on the ending of genitive singular: `aa` for those with the ending *-a*, `ae` for those ending in *-e*, and `ay` for those ending in *-y*.

Some masculine animate nouns have adjective-like declension (but no negation, no gradation). Most of them can or could form the feminine derivative, but some (like the noun *mužský*, used as colloquial form of *manžel* (husband)) cannot. This distinction leads to two other classes, `af` for those masculine animate nouns with adjective-like declension and feminine forms, and `a` for those without feminine forms.

Further, proper masculine names (with the exception of family names of foreign origin, which are excluded from the model at present, and also with

the exception of the names of nation members, see later) behave in a little different way, especially in nominative plural, even if they end similarly as the other general masculine animate nouns. On the other hand, their paradigm can be recognized using the nominative singular form only; this allows to use only one new class, namely, `ja`.

Inside the proper names group, a smaller group of names of nation or state members forms a different subgroup. They can, of course, form the feminine derivative, and other differences are encountered in the nominative plural forms. The class `na` is to be assigned to such names.

Masculine inanimate nouns are the most complicated group. The important forms for the assignment of the paradigm are (except the usual nominative singular) the genitive singular (four possibilities: *-u, -a*, variation *-u/-a, -e*), locative singular (four possibilities: *-u, -ě(e)*, variation *-u/-ě(e), -i*) and locative plural (four possibilities: *-ech, -ách, -ích*, variation *-ách/-ích*). Fortunately, not all combinations (which would form 64 classes and several hundred paradigms) are realized. As we consider also colloquial forms of the locative plural, the ending(s) of the locative plural can be determined by the other forms (*Nsg, Gsg* and *Lsg*); this restrict the number of possible classes to 16, out of which only the following ten materialize:

class name	Gsg	Lsg
`iuu`	*-u*	*-u*
`iau`	*-a*	*-u*
`iuau`	*-u/-a*	*-u*
`iue`	*-u*	*-ě(e)*
`iae`	*-a*	*-ě(e)*
`iuae`	*-u/-a*	*-ě(e)*
`iuue`	*-u*	*-u/-ě(e)*
`iaue`	*-a*	*-u/-ě(e)*
`iuaue`	*-u/-a*	*-u/-ě(e)*
`ie`	*-ě(e)*	*-i*

Out of these, `iuu`, `iuue` and `ie` are the most frequent in Czech.

Further, some nouns of Greek or Latin origin have to be distinguished. They are always of the `iuu` type, but differ in the nominative singular, where they attach *-us*, *-os* or *-es*. As some of the original Czech nouns have the same endings, the foreign ones have to be assigned a special class, `iuuc`.

Further, for the adjective-like declension (e.g. *rýnský* (Rhenish gulden)), class `i` has to be introduced.

Generally, all the classes introduced so far could each be expanded to two more classes, namely, one for singularia tantum (or nouns met only in singular) and one for pluralia tantum (or nouns met only in plural).

In fact, however, this is necessary only for some classes:

class(es)	derived by restriction of
`ns,np`	`n`
`fp`	`f`
`is`	`i`
`ip`	all `i`-based classes; no ambiguities when assigning the paradigm
`aes,aep`	`ae` (only few nouns + their compounds)
`ieep`	special class (similar, but not equal to `iep`, which would be derived from `ie`)

5.1.4 The Description of Classes

The detailed description of the individual classes is given in the Appendices, in the form of an actual computer listing to avoid misprints during retyping. Here we present a brief description of both Appendix A and B.

In the Appendix A, the individual paradigms are in the form

```
<paradigm name> <list of endings>
```

where the list of endings possibly extends to more lines (the continuation lines being marked by space in the first column). The `<list of endings>` simply lists all the possible (different) endings, the information about the particular morphological category(-ies) belonging to the individual endings being elsewhere.

In the Appendix B, the rules for paradigm assignment inside the individual classes are presented. The class is introduced by `>>`, its name and fullstop, always on a separate line.

Each rule has the form

```
<paradigm name> :<cancel count><list of end segments>.
```

and it should be read as

> "if the end segment of the nominative singular (= dictionary) form of the word in question, in the appropriate length, matches one of the segments in the <list of end segments>, and no longer end segment from the word in question matches any segment from any <list of end segments> of the rules inside the given class, then
>
> 1. cut away <cancel count> characters from the end of the dictionary form and store the result as the stem;
> 2. assign the <paradigm name> as the resulting paradigm (set of endings)."

The individual forms can then be constructed according to the rule given in the section 5.1.1 by simple concatenation of the stem and the endings from the <list of endings> of the assigned <paradigm name> (see Appendix A and the description earlier in this section).

5.1.5 Implementation and Conclusions

For the interactive system, see Chapter 5.2.

In the batch mode, the system simply reads the input words (which have to be in the dictionary form, i.e., nominative singular) together with manually assigned classes and produces stems together with assigned paradigm names.

An implementation of this system on an IBM PC was used for creating the noun subdictionary for the first, linguistically based, commercially available spelling checker for Czech (for PC, of course). It was slightly modified (e.g., the pure Appendix A data was used, disregarding the morphological categories) and simplified (e.g., the classes `iae, iaue, iuae` and `iuaue` collapsed into one, because the ending *-u* is used in the dative singular for all masculine inanimates of this type).

The data are stored in the form <word in Nsg> <class>, then they are processed to obtain <stem> <paradigm name> for the purpose of creating

a new version of the main spelling dictionary. This way, all the almost 30.000 regular nouns were entered and are still maintained, which proves the usability of the system.

Practical experience with the system has shown that some improvements have to be made. First, when considering the derivations (such as feminine forms or possessives), one stem + paradigm pair is often not enough. An approach similar to (Weisheitelová, Králíková and Sgall 1982), i.e., the possibility to assign more than one stem + paradigm pair by the rules, will solve this (actually, it has been solved in the interactive system of word classification, see chapter 5.2). Further, still new rules have to be added, together with new paradigm definitions, to cover all possible Czech texts (e.g., almost none of family names of foreign origin, such as *Jaruzelski*, is treated properly in the rules presented here). And, of course, it would be nice to extend the algorithms to cover adjectives and productive verb classes, too, which is basically easy, but problems arise with some derivations, especially between nouns and adjectives with complete paradigms, because of negation and gradation.

5.2 Interactive Word Classification

5.2.1 Introduction

It is obvious that even when using some kind of semi-automatic assignment of classes one wants to check and/or change things directly in the resulting dictionary. Also, once some system really works, an enormous number of unrecognized words keeps coming in and not surprisingly, some of them are real omissions which have to classified and added to the dictionary. Therefore, an interactive tool has been developed for adding and editing the source dictionary, which uses some knowledge about word inflections to minimize the effort needed for correct word class assignment.

5.2.2 Mode of Operation

The typical use of the interactive tool (called `MCLASS`) consists of making a list of "unknown" words (actually, word forms – they need not be in the usual dictionary form), one per line, and then running `MCLASS` on them. For each

word form in the list, the program MCLASS asks some simple multiple–choice questions, mainly of the type "*Which phrase is correct Czech: (a) ... (b) .... (c) ...?*". Depending on the answer, it either poses another question, or displays a suggestion of a dictionary (base) form and a class, together with a list of all possible forms. If confirmed by the operator, the entry is added (or changed) in the source dictionary. Of course, it is possible to see all existing entries, modify them, delete them, and even add new entries. It is also possible to add or modify the word classes "by hand". In such a case, the automatic assignment procedure is skipped. Only the list of all forms is displayed and a confirmation is requested.

5.2.3 The Class Assignment Information

The MCLASS program differs from the batch version (see 5.1.1) in several ways:

- it is interactive and its design heavily uses this fact;
- it can handle adjectives and verbs as well;
- it can handle any input form, not only the dictionary forms (nom. sg. for nouns and adjectives, infinitive for verbs);
- it asks the operator only when needed; for some word forms, it only ask for confirmationů
- the format of the classification data is different, as it must handle questions and answers, in a multiple-step process.

Chapter 6

Future Directions

Currently, work is in progress on a tagger and searching tools, which should enable flexible use of the tagged corpora. Czech was chosen to start with in the tagger project, as it is interesting to see which approach to tagging is going to be the most successful: it is clear that for such a inflective language as Czech certainly is (its vocabulary contains over 5 million forms, and the length of the full flat tag list is almost 3 thousand) the "classical" statistical approach to tagging is not quite the right thing to do, although we would like to test it anyway on a subset of the language. Meanwhile we are working on new methods to deal with the usual problems arising from having too many parameters.

The morphology software itself needs a better parser, otherwise we do not plan to develop the formalism nor the software any further.

References

Aho, A.V., Hopcroft, J.E., Ullman, J.D. (1976): The Design and Analysis of Computer Algorithms. Addison–Wesley, Reading MA

Aho, A.V., Ullman, J.D. (1972): The Theory of Parsing, Translation and Compiling, Vol 1. Parsing. Prentice Hall, Englewood Cliffs NJ

Bělič, J. (ed.) (1981): Pravidla českého pravopisu. Státní pedagogické nakladatelství, Prague

Bémová, A., Králíková, K. (1988): K otázkám automatického zpracování českého tvarosloví. Slovo a Slovesnost **XLIX/4**, 285–295

Booij, G., van Marle, J. (eds.) (1993): Yearbook of Morphology 1992. Kluwer Academic Publishers

Borin, L. (1991): The automatic induction of morphological regularities. Dept. of Linguistics, University of Uppsala, Uppsala. PhD Thesis

Borota, J., Hajič, J. (1984): Metoda MOZAIKA (The Method MOSAIC). Proceedings of the Moderní programování (Modern programming) **3**, 75–83. Trutnov, Czechoslovakia, May 27th–June 4th 1984

Brown, P.F., Della Pietra, S.A., Della Pietra, V.J., Goldsmith, M.J., Hajič, J., Mercer, R.L., Mohanty, S. (1993): But Dictionaries are Data Too. Proceedings of the HLT, 1993

Cejpek, J. (1982): Automatické vyhladávanie informácií z úplného textu. Metoda SIUT.. Slovenská technická knižnica, Bratislava

Čermák, F., Králík, J., Pala, K. (1992): Počítačová lexikografie a čeština (Počítačový fond češtiny). Slovo a Slovesnost **LIII**, 41–43

Chytil, M. (1984): Automaty a gramatiky. (Matematický seminář, vol. 19) SNTL, Prague

Daneš, F., Grepl, M., Hlavsa, Z. (eds.) (1987): Mluvnice češtiny 3 (Skladba). Academia, Prague

Dokulil, M., Horálek, K., Hůrková, J., Knappová, M., Petr, J. (eds.) (1986): Mluvnice češtiny 1 (Fonetika etc.). Academia, Prague

Drózd, J., Hajič, J. (1990): Kontrola českého pravopisu na PC (Czech spelling-checker on a PC). Proceedings of the AI'90. ČSVTS Prague, April 12th–14th, 1990

Filipec, J., Daneš, F. (eds.) (1978): Slovník spisovné češtiny pro školu a veřejnost. Academia, Prague

Finkler, W., Neumann, G. (1988): Morphix: A Fast Realization of a Classification-Based Approach to Morphology. Universität des Saarlandes, KI-Labor am Lehrstuhl für Informatik IV, Saarbrücken. Bericht

Gazdar, G. (1985): Finite State Morphology: A Review of Koskenniemi (1983). Linguistics **23**, 597–607

Hajič, J. (1980): Jednoduchý systém kontaktu s bází dat v češtině (A System of Querying a Database in Czech). Československá informatika **12**, 334–338

Hajič, J. (1983): KODAS - A Natural Language Interface to a Simple Database. The Prague Bulletin of Mathematical Linguistics **39**, 65–75

Hajič, J. (1984a): KODAS - A Simple Method of Natural Language Interface to a Database. (Explizite Beschreibung der Sprache und automatische Textbearbeitung, vol. XI) Charles University, Prague

Hajič, J. (1984b): Softwarový systém pro analýzu českých pokynů řídících robota (A Software System for an Analysis of Robot Controlling Commands in Czech). Dept. of CS, Faculty of Mathematics and Physics, Charles University, Prague. Thesis. 195 pp.

Hajič, J. (1985): Programming a Simple Natural Language Interface (Part 1). The Prague Bulletin of Mathematical Linguistics **43**, 37–62

Hajič, J. (1986): Programming a Simple Natural Language Interface (Part 2). The Prague Bulletin of Mathematical Linguistics **44**, 23–56

Hajič, J. (1987a): RUSLAN - an I/O Overview. Proceedings of the AI'87. ČSVTS Prague, April 1987

Hajič, J. (1987b): An MT System for Closely Related Languages. Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics. Copenhagen, April 1st–3rd 1987

Hajič, J. (1987c): Morfologie pro flexívní a aglutinační jazyky (The Morphology for Flexive and Agglutinative Languages). Proceedings of the SOFSEM'87. Malenovice, Beskydy, ÚVT UJEP Brno, November 29th–December 11th, 1987

Hajič, J. (1988a): NALCOM: A Multilevel NL-interface. (Explizite Beschreibung der Sprache und automatische Textbearbeitung, vol. XV) Charles University, Prague. 146 pp.

Hajič, J. (1988b): Formální morfologie (Formal Morphology). Proceedings of the AI'88. ČSVTS Prague, March 8th–10th, 1988

Hajič, J. (1988c): Formal Morphology. Proceedings of the 12th International Conference on Computational Linguistics. Budapest, August 22nd–27th 1988

Hajič, J. (1989a): Morphotactics by Attribute Grammar. Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics. Manchester, April 10th–12th 1989

Hajič, J. (1989b): Strojový překlad v Japonsku (Machine translation activities in Japan). Proceedings of the AI'89. ČSVTS Prague, April 24th–26th, 1989

Hajič, J. (1992): Semi-automatic Classification of Czech Nouns. In: The Prague Studies in Mathematical Linguistics. Prague

Hajič, J. (1993): Multilanguage Morphology with Broad Coverage. Proceedings of the SwAN'21. ISSCO, Geneva, September 5th–9th, 1993

Hajič, J., Drózd, J. (1990): Spelling-checking for Highly Inflective Languages. Proceedings of the 13th International Conference on Computational Linguistics **III**, 358–360. Helsinki, August 1990

Hajič, J., Hajičová, E. (1980): Jednoduchý systém kontaktu s bází dat v češtině (A System of Querying a Database in Czech). Proceedings of the SOFSEM'80. Bílý Kříž, November 23th–December 5th, 1980

Hajič, J., Kirschner, Z., Rosen, A. (1986): Některé problémy aplikací strojového překladu (On Some Problems of Machine Translation Applications). Proceedings of the Využití lingvistických přístupů v informatice (Using the Linguistic Approach in Information Retrieval) , Prague. June 9th–10th 1986

Hajič, J., Oliva, K. (1986): Projekt česko–ruského strojového překladu (The Czech To Russian Machine Translation Project). Proceedings of the SOFSEM'86. Liptovský Ján, November 23th–December 5th 1986

Hajič, J., Rosen, A., Hajičová, E. (1992): Machine Translation Research in Czechoslovakia. (META, vol. 37, no. 4) Les Presses de l'Université de Montréal

Hajič, J., Rosen, A., Hajičová, E., Skoumalová, H. (1992): MATRACE. Proceedings of the The IBM AI conference, Prague. November 18th–19th 1992

Hajič, J., Zvelebil, V. (1986): Komunikační interface v češtině (Communication Interface in Czech). Proceedings of the Využití lingvistických přístupů v informatice (Using the Linguistic Approach in Information Retrieval) , Prague. June 9th–10th 1986

Hajičová, E., (ed.) (1993): projekt MATRACE. Prague. výroční zpráva pro GA ČR, 1993

Havránek, B. (ed.) (1989): Slovník spisovného jazyka českého. Academia, Prague

Havránek, B., Jedlička, A. (1963): Česká mluvnice. Státní pedagogické nakladatelství

Hellberg, S. (1974): Graphonomic Rules in Phonology. (Nordistica Gothoburgensia, no. 7) Acta Universitatis Gothoburgensis, Göteborg

Jirků, P., Hajič, J. (1982): Inferencing and search for an answer in TIBAQ. Abstracts of COLING'82. Prague, July 5th–10th, 1982

Karlsson, F. (ed.) (1985): Computational Morphosyntax (Report on Research 1981). (Publications, no. 13) University of Helsinki, Dept. of General Linguistics, Helsinki

Karlsson, F. (1986): A paradigm-based morphological analyzer. Papers from the Fifth Scandinavian Conference of Computational Linguistics, Helsinki

Karlsson, F., Koskenniemi, K. (1985): A Process Model of Morphology and Lexicon. Folia Linguistica **XIX/1-2**, 207–231

Kay, M. (1987): Nonconcatenative Finite-State Morphology. Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics. Copenhagen, April 1st–3rd 1987

Kirschner, Z. (1982a): Experiment s metodou úplného textu. Československá informatika **24**, 105–112

Kirschner, Z. (1982b): A Dependency Based Analysis of English for the Purpose of Machine Translation. (Explizite Beschreibung der Sprache und automatische Textbearbeitung, vol. IX) Charles University, Prague

Kirschner, Z. (1983): MOSAIC – A Method of Automatic Extraction of Significant Terms from Texts. (Explizite Beschreibung der Sprache und automatische Textbearbeitung, vol. X) Charles University, Prague

Klein, W. (ed.) (1988): (Linguistics, vol. 26-4) Mounton de Gruyter, Berlin New York Amsterdam

Knuth, D.E. (1969): The Art of Computer Programming, Vol. I-III. Addison–Wesley, Reading MA

Koskenniemi, K. (1983): Two-level Morphology: a General Computational Model for Word-form Recognition and Production. (Publications, no. 11) Department of General Linguistics, University of Helsinki, Helsinki

Laciga, Z. (1993): ASPI - Automatizovaný systém právních informací (ASPI - An Automated System of Legal Information). MAA **5**

Machová, S. (1977): Functional Generative Description of Czech. (Explizite Beschreibung der Sprache und automatische Textbearbeitung, vol. III) Charles University, Prague

Martincová, Z. (ed.) (1993): Pravidla českého pravopisu. Pansofia, Prague
Matthews, P.H. (1982): Morphology. (Cambridge Textbooks in Linguistics) Cambridge University Press, Cambridge
Oliva, K. (1983): Automatická syntaktická analýza češtiny. Faculty of Mathematics and Physics, Prague. Thesis
Osolsobě, K., Pala, K. (1990): Czech Stem Dictionary for IBM PC XT/AT. Proceedings of the Conference on Computer Lexicography, Balatonfüred. September 1990
Osolsobě, K., Pala, K., Franc, S. (1987): Česká morfologie a syntax v Prologu. Proceedings of the Sofsem'87, Bratislava
Pala, K. (1992): Počítačové zpracování češtiny. Filozofická fakulta Masarykovy univerzity, Brno. habilitační práce
Panevová, J. (1979): Transducing Components of Functional Generative Description 1 (From Tectogrammatics to Morphemics). (Explizite Beschreibung der Sprache und automatische Textbearbeitung, vol. IV) Charles University, Prague
Panevová, J. (1981): Lexical Input Data for Experiments with Czech. (Explizite Beschreibung der Sprache und automatische Textbearbeitung, vol. VI) Charles University, Prague
Panevová, J., Cejpek, J., Zvelebil, V. (1988): ASIMUT 2 - Metoda vyhledávání v úplných textech. Československá informatika **30**, 294–299
Panevová, J., Králíková, K. (1990): ASIMUT - A Method for Automatic Information Retrieval from Full Texts. (Explizite Beschreibung der Sprache und automatische Textbearbeitung, vol. XVII) Charles University, Prague
Petr, J. (ed.) (1986): Mluvnice češtiny 2 (Tvarosloví). Academia, Prague
Poldauf, I. (ed.) (1986): Česko-anglický slovník. Státní pedagogické nakladatelství, Prague
Ritchie, G.D. (1992): Languages Generated by Two-Level Morphological Rules. Computational Linguistics **18**, 41–59
Ritchie, G.D., Russell, G.J., Black, A.W., Pulman, S.G. (1992): Computational Morphology: Practical Mechanisms for the English Lexicon. MIT Press, Cambridge, MA
Sågvall, A-L. (1973): A System for Automatic Inflectional Analysis (Implemented for Russian). (Data linguistica, vol. 8) Almqvist & Wiksell Informationsindustri AB, Uppsala
Salomaa, A. (1973): Formal Languages. Academic Press, New York NY

Sgall, P. (1960): Soustava pádových koncovek v češtině. AUC - Slavica Pragensia

Sgall, P. (1967): Generativní popis jazyka a česká deklinace. (Studie a práce lingvistické, vol. 6) Academia, Prague

Sgall, P., Bémová, A., Hajičová, E., Hajičová, I., Jirků, P., Panevová, J., Pitha, P., Plátek, M., Vrbová, J. (1986): Úvod do syntaxe a sémantiky. (Studie a práce lingvistické, vol. 22) Academia

Sgall, P., Hajič, J. (1984): Kontakt s bází dat v češtině (Querying a Database in Czech). In: Využitie lingvistických metód vo VTEI (The Use of Linguistic Methods in Information Services). Slovak Technical Society, Bratislava, pp. 33–65

Sgall, P., Hajičová, E., Panevová, J. (1986): The Meaning of a Sentence in its Semantic and Pragmatic Aspects. Academia - North–Holland, Prague - Amsterdam

Sgall, P., Weisheitelová, J. (1968): K deklinaci českých substantiv ženského rodu. AUC - Slavica Pragensia

Sgall, P., Zima, M. (1986): Questions of Orthography and Transliteration. (Explizite Beschreibung der Sprache und automatische Textbearbeitung, vol. XII) Charles University, Prague

Shieber, S.M. (1986): An Introduction to Unification-Based Approaches to Grammar. CSLI, Stanford

Slavíčková, E. (1975): Retrográdní morfematický slovník češtiny. Academia, Prague

Šmilauer, V. (1969): Novočeská skladba. Státní pedagogické nakladatelství, Prague

Sproat, R. (1992): Morphology and Computation. (ACL-MIT Press Series in Natural Language Processing) MIT Press

Těšitelová, M. (1983): Psaná a mluvená odborná čeština z kvantitativního hlediska (v rámci věcného stylu). (Linguistica, vol. IV) Academia, Prague

Těšitelová, M. (ed.) (1985): Kvantitativní charakteristiky současné češtiny. (Studie a práce lingvistické, vol. 19) Academia, Prague

Těšitelová, M. (1987): O češtině v číslech. (Malá jazyková knižnice, vol. 4) Academia, Prague

Těšitelová, M. (1992): Quantitative Linguistics. Academia, Prague

Těšitelová, M., Petr, J., Králík, J. (1985): Retrográdní slovník tvarů adjektiv v současné češtině. Ústav pro jazyk český ČSAV, Prague

Těšitelová, M., Petr, J., Králík, J. (1986): Retrográdní slovník současné češtiny. Academia, Prague

Trnková, A. (1983): Cvičebnice české morfologie pro zahraniční studenty (I. Formální morfologie). Státní pedagogické nakladatelství, Prague. Skripta UK Praha

Trost, H. (1990): The application of two-level morphology to non-concatenative German morphology. Proceedings of the 13th International Conference on Computational Linguistics. Helsinki, August 1990

Warwick, S., Hajič, J., Russell, G. (1990): Searching on Tagged Corpora: Linguistically Motivated Concordance Analysis. Electronic Text Research: Proceedings of the Sixth Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research. University of Waterloo, 28th–30th October 1990

Warwick-Armstrong, S., Winarske, A., Russell, G., Hajič, J. (1992): Tagging and Alignment of Parallel Texts: Current Status of BCP. Proceedings of the 3rd Conference on Applied Natural Language Processing. Trento, March 31st–April 3rd, 1992

Weisheitelová, J., Bémová, A. (1979): Transducing Components of Functional Generative Description 2. (Explizite Beschreibung der Sprache und automatische Textbearbeitung, vol. V) Charles University, Prague

Weisheitelová, J., Králíková, K., Sgall, P. (1982): Morphemic Analysis of Czech. (Explizite Beschreibung der Sprache und automatische Textbearbeitung, vol. VII) Charles University, Prague

Zwicky, A.M., Wallace, R.E. (eds.) (1984): Papers on Morphology. (Working Papers in Linguistics, no. 29) Dept. of Linguistics, The Ohio State University, Columbus

Appendix A

Nouns–sets of endings

pn1 0,a,u,ovi,em,e,ů,ům,y,ech,i,ové,ův,ova,ovo,ovu,ovy,ově,
ovou,ovým,ových,ovými,ovýma

pn2 0,a,u,ovi,em,e,ů,ům,y,ech,i,ův,ova,ovo,ovu,ovy,ově,ovou,
ovým,ových,ovými,ovýma

pn2j 0,a,u,ovi,em,e,ů,ům,y,ech,i,ův,ova,ovo,ovu,ovy,ově,ovou,
ovým,ových,ovými,ovýma,ka,ky,ce,ku,kou,ko,ek,kám,kách,kami,
čin,čina,čino,činu,činy,čině,činou,činým,čini,činých,činými,
činýma

pn3 0,a,u,ovi,em,e,ů,ům,y,ech,ové,ův,ova,ovo,ovu,ovy,ově,ovou,
ovým,ových,ovými,ovýma

pn3j 0,a,u,ovi,em,e,ů,ům,y,ech,ové,ův,ova,ovo,ovu,ovy,ově,
ovou,ovým,ových,ovými,ovýma,ka,ky,ce,ku,kou,ko,ek,kám,kách,
kami,čin,čina,čino,činu,činy,čině,činou,činým,čini,činých,
činými,činýma

pn4j 0,a,u,ovi,em,e,ů,ům,y,ech,é,ův,ova,ovo,ovu,ovy,ově,ovou,
ovým,ových,ovými,ovýma,ka,ky,ce,ku,kou,ko,ek,kám,kách,kami,
čin,čina,čino,činu,činy,čině,činou,činým,čini,činých,činými,
činýma

pn5 0,a,u,ovi,em,e,ů,ům,y,ech,i,é,ův,ova,ovo,ovu,ovy,ově,ovou,
ovým,ových,ovými,ovýma

pn6 r,ra,ru,rovi,rem,ře,rů,rům,ry,rech,ři,rův,rova,rovo,rovu,
rovy,rově,rovou,rovým,rových,rovými,rovýma

pn6r r,ra,ru,rovi,rem,re,rů,rům,ry,rech,ři,rův,rova,rovo,rovu,
rovy,rově,rovou,rovým,rových,rovými,rovýma

```
pn6f r,ra,ru,rovi,rem,re,rů,rům,ry,rech,ři,rův,rova,rovo,rovu,
  rovy,rově,rovou,rovým,rových,rovými,rovýma,rka,rky,rce,rku,
  rkou,rko,rek,rkám,rkách,rkami,rčin,rčina,rčino,rčinu,rčiny,
  rčině,rčinou,rčiným,rčini,rčiných,rčinými,rčinýma
pn6fy r,ra,ru,rovi,rem,ře,rů,rům,ry,rech,ři,rův,rova,rovo,
  rovu,rovy,rově,rovou,rovým,rových,rovými,rovýma,ryně,ryni,
  ryní,ryň,ryním,ryních,rynětmi,rynin,rynina,rynino,ryninu,
  ryniny,rynině,ryninou,ryniným,rynini,ryniných,ryninými, ryninýma
pn7 es,a,u,ovi,em,e,ové,ů,ům,y,ech,ův,ova,ovo,ovu,ovy,ově,
  ovou,ovým,ových,ovými,ovýma
pn8 es,sa,su,sovi,sem,se,si,sů,sům,sy,sech,sův,sova,sovo,sovu,
  sovy,sově,sovou,sovým,sových,sovými,sovýma
pn9 ev,va,vu,vovi,vem,ve,vi,vů,vům,vy,vech,vův,vova,vovo,vovu,
  vovy,vově,vovou,vovým,vových,vovými,vovýma
pn11 el,la,lu,lovi,lem,le,lové,lů,lům,ly,lech,lův,lova,lovo,
  lovu,lovy,lově,lovou,lovým,lových,lovými,lovýma
pn12 us,a,u,ovi,em,e,ové,ů,ům,ích,i,ův,ova,ovo,ovu,ovy,ově,
  ovou,ovým,ových,ovými,ovýma
pn13 os,a,u,ovi,em,e,ové,ů,ům,y,ech,ův,ova,ovo,ovu,ovy,ově,
  ovou,ovým,ových,ovými,ovýma
pn16 k,ka,ku,kovi,kem,ci,ků,kům,ky,cích,kův,kova,kovo,kovu,
  kovy,kově,kovou,kovým,kových,kovými,kovýma
pn16ik ík,íka,íku,íkovi,íkem,íci,íků,íkům,íky,ících,íkův,
  íkova,íkovo,íkovu,íkovy,íkově,íkovou,íkovým,íkových,íkovými,
  íkovýma,ice,ici,icí,icím,icích,icemi,ic,ičin,ičina,ičino,
  ičinu,ičiny,ičině,ičinou,ičiným,ičini,ičiných,ičinými,ičinýma
pn16j k,ka,ku,kovi,kem,kové,ků,kům,ky,cích,kův,kova,kovo,kovu,
  kovy,kově,kovou,kovým,kových,kovými,kovýma
pn17j ek,ka,ku,kovi,kem,kové,ků,kům,ky,cích,kův,kova,kovo,
  kovu,kovy,kově,kovou,kovým,kových,kovými,kovýma
pn17 ek,ka,ku,kovi,kem,ci,kové,ků,kům,ky,cích,kův,kova,kovo,
  kovu,kovy,kově,kovou,kovým,kových,kovými,kovýma
pn18 k,ka,ku,kovi,kem,ci,kové,ků,kům,ky,cích,kův,kova,kovo,
  kovu,kovy,kově,kovou,kovým,kových,kovými,kovýma
pn19 g,ga,gu,govi,gem,gové,gů,gům,gy,zích,gův,gova,govo,govu,
  govy,gově,govou,govým,gových,govými,govýma
```

```
pn20 h,ha,hu,hovi,hem,zi,hové,hů,hům,hy,zích,hův,hova,hovo,
  hovu,hovy,hově,hovou,hovým,hových,hovými,hovýma
pn21 h,ha,hu,hovi,hem,hové,hů,hům,hy,zích,hův,hova,hovo,hovu,
  hovy,hově,hovou,hovým,hových,hovými,hovýma
pn22 ch,cha,chu,chovi,chem,ši,chů,chům,chy,ších,chův,chova,
  chovo,chovu,chovy,chově,chovou,chovým,chových,chovými,chovýma
pn23 ch,cha,chu,chovi,chem,chové,chů,chům,chy,ších,chův,chova,
  chovo,chovu,chovy,chově,chovou,chovým,chových,chovými,chovýma
pn24 ch,cha,chu,chovi,chem,ši,chové,chů,chům,chy,ších,chův,
  chova,chovo,chovu,chovy,chově,chovou,chovým,chových,chovými,
  chovýma
pn26 0,a,u,ovi,em,e,ův,ova,ovo,ovu,ovy,ově,ovou,ovým,ových,
  ovými,ovýma
pn27 něk,ňka,ňku,ňkovi,ňkem,ňků,ňkům,ňky,ňcích,ňci,ňkové,ňkův,
  ňkova,ňkovo,ňkovu,ňkovy,ňkově,ňkovou,ňkovým,ňkových,ňkovými,
  ňkovýma
pn27j něk,ňka,ňku,ňkovi,ňkem,ňků,ňkům,ňky,ňcích,ňkové,ňkův,
  ňkova,ňkovo,ňkovu,ňkovy,ňkově,ňkovou,ňkovým,ňkových,ňkovými,
  ňkovýma
pn28j děk,ďka,ďku,ďkovi,ďkem,ďků,ďkům,ďky,ďcích,ďkové,ďkův,
  ďkova,ďkovo,ďkovu,ďkovy,ďkově,ďkovou,ďkovým,ďkových,ďkovými,
  ďkovýma
pn29j těk,ťka,ťku,ťkovi,ťkem,ťků,ťkům,ťky,ťcích,ťkové,ťkův,
  ťkova,ťkovo,ťkovu,ťkovy,ťkově,ťkovou,ťkovým,ťkových,ťkovými,
  ťkovýma
pn5j 0,a,u,ovi,em,e,ů,ům,y,ech,i,é,ův,ova,ovo,ovu,ovy,ově,
  ovou,ovým,ových,ovými,ovýma,ka,ky,ce,ku,kou,ko,ek,kám,kách,
  kami,čin,čina,čino,činu,činy,čině,činou,činým,čini,činých,
  činými,činýma
pno o,a,u,ovi,em,e,ové,ů,ům,y,ech,ův,ova,ovo,ovu,ovy,ově,ovou,
  ovým,ových,ovými,ovýma
pnok ko,ka,ku,kovi,kem,kové,ků,kům,ky,cích,kův,kova,kovo,kovu,
  kovy,kově,kovou,kovým,kových,kovými,kovýma
pnoh ho,ha,hu,hovi,hem,hové,hů,hům,hy,zích,hův,hova,hovo,hovu,
  hovy,hově,hovou,hovým,hových,hovými,hovýma
pnog go,ga,gu,govi,gem,gové,gů,gům,gy,zích,gův,gova,govo,govu,
```

```
  govy,gově,govou,govým,gových,govými,govýma
pnoc o,a,u,ovi,em,e,ové,ů,ům,ích,i,ův,ova,ovo,ovu,ovy,ově,
  ovou,ovým,ových,ovými,ovýma
pne e,a,u,ovi,em,ové,ů,ům,y,ech,ův,ova,ovo,ovu,ovy,ově,ovou,
  ovým,ových,ovými,ovýma
pnus us,a,u,ovi,em,e,ův,ova,ovo,ovu,ovy,ově,ovou,ovým,ových,
  ovými,ovýma
pd1 a,y,ovi,u,ou,o,ů,ům,ech,ové,ův,ova,ovo,ovu,ovy,ově,ovou,
  ovým,ových,ovými,ovýma
pd2f a,y,ovi,u,ou,o,ů,ům,ech,i,é,ův,ova,ovo,ovu,ovy,ově,ovou,
  ovým,ových,ovými,ovýma,ka,ky,ce,ku,kou,ko,ek,kám,kách,kami,
  čin,čina,čino,činu,činy,čině,činou,činým,čini,činých,činými,
  činýma
pd2 a,y,ovi,u,ou,o,ů,ům,ech,i,é,ův,ova,ovo,ovu,ovy,ově,ovou,
  ovým,ových,ovými,ovýma
pd3 a,y,ovi,u,ou,o,ů,ům,ech,i,ové,ův,ova,ovo,ovu,ovy,ově,ovou,
  ovým,ových,ovými,ovýma
pd4 a,i,ovi,u,ou,o,ové,ů,ům,ích,ův,ova,ovo,ovu,ovy,ově,ovou,
  ovým,ových,ovými,ovýma
pd4d ďa,di,ďovi,ďu,ďou,ďo,ďové,ďů,ďům,dích,ďův,ďova,ďovo,ďovu,
  ďovy,ďově,ďovou,ďovým,ďových,ďovými,ďovýma
pd4t ťa,ti,ťovi,ťu,ťou,ťo,ťové,ťů,ťům,tích,ťův,ťova,ťovo,ťovu,
  ťovy,ťově,ťovou,ťovým,ťových,ťovými,ťovýma
pd4n ňa,ni,ňovi,ňu,ňou,ňo,ňové,ňů,ňům,ních,ňův,ňova,ňovo,ňovu,
  ňovy,ňově,ňovou,ňovým,ňových,ňovými,ňovýma
pd5 ka,ky,kovi,ku,kou,ko,kové,ků,kům,cích,kův,kova,kovo,kovu,
  kovy,kově,kovou,kovým,kových,kovými,kovýma
pd6 ga,gy,govi,gu,gou,go,gové,gů,gům,zích,gův,gova,govo,govu,
  govy,gově,govou,govým,gových,govými,govýma
pd7 ha,hy,hovi,hu,hou,ho,hové,hů,hům,zích,hův,hova,hovo,hovu,
  hovy,hově,hovou,hovým,hových,hovými,hovýma
pd8 cha,chy,chovi,chu,chou,cho,chové,chů,chům,ších,chův,chova,
  chovo,chovu,chovy,chově,chovou,chovým,chových,chovými,chovýma
mdmih hý,hého,hému,hém,hým,hých,hými,hé
mdmi ý,ého,ému,ém,ým,í,ých,é,ými
```

```
mdmik ký,kého,kému,kém,kým,kých,kými,ké
mdmick cký,ckého,ckému,ckém,ckým,ckých,ckými,cké
mdmisk ský,ského,skému,ském,ským,ských,skými,ské
mdmich chý,chého,chému,chém,chým,chých,chými,ché
mdmir rý,rého,rému,rém,rým,rých,rými,ré
mdmah hý,hého,hému,hém,hým,hých,hými,hé,zí
mdma ý,ého,ému,ém,ým,í,ých,é,ými
mdmak ký,kého,kému,kém,kým,kých,kými,ké,cí
mdmack cký,ckého,ckému,ckém,ckým,ckých,ckými,cké,čtí
mdmask ský,ského,skému,ském,ským,ských,skými,ské,ští
mdmach chý,chého,chému,chém,chým,chých,chými,ché,ší
mdmar rý,rého,rému,rém,rým,rých,rými,ré,ří
mdsh hý,hého,hému,hém,hým,zí,há,hých,hými,hé
mds ý,ého,ému,ém,ým,í,á,é,ých,ými
mdsk ký,kého,kému,kém,kým,cí,ká,kých,kými,ké
mdsck cký,ckého,ckému,ckém,ckým,čtí,cká,ckých,ckými,cké
mdssk ský,ského,skému,ském,ským,ští,ská,ských,skými,ské
mdsch chý,chého,chému,chém,chým,ší,chá,chých,chými,ché
mdsr rý,rého,rému,rém,rým,ří,rá,rých,rými,ré
mdafh hý,hého,hému,hém,hým,zí,há,hých,hými,hé,hou
mdaf ý,ého,ému,ém,ým,í,á,é,ých,ými,ou
mdafk ký,kého,kému,kém,kým,cí,ká,kých,kými,ké,kou
mdafck cký,ckého,ckému,ckém,ckým,čtí,cká,ckých,ckými,cké,ckou
mdafsk ský,ského,skému,ském,ským,ští,ská,ských,skými,ské,skou
mdafch chý,chého,chému,chém,chým,ší,chá,chých,chými,ché,chou
mdafr rý,rého,rému,rém,rým,ří,rá,rých,rými,ré,rou
mz1 0,e,i,ovi,em,ů,ům,ích,ové,ův,ova,ovo,ovu,ovy,ově,ovou,
  ovým,ových,ovými,ovýma
mz1i 0,e,i,ovi,em,ů,ům,ích,ův,ova,ovo,ovu,ovy,ově,ovou,ovým,
  ových,ovými,ovýma
mz1e 0,e,i,ovi,em,ů,ům,ích,é,ův,ova,ovo,ovu,ovy,ově,ovou,ovým,
  ových,ovými,ovýma,ka,ky,ce,ku,kou,ko,ek,kám,kách,kami,čin,
  čina,čino,činu,činy,čině,činou,činým,čini,činých,činými, činýma
```

mz2 ď,dě,di,ďovi,děm,ďů,ďům,dích,ďův,ďova,ďovo,ďovu,ďovy,ďově,
ďovou,ďovým,ďových,ďovými,ďovýma

mz2o ď,dě,di,ďovi,děm,ďů,ďům,dích,ďové,ďův,ďova,ďovo,ďovu,
ďovy,ďově,ďovou,ďovým,ďových,ďovými,ďovýma

mz3 ť,tě,ti,ťovi,těm,ťů,ťům,tích,ťův,ťova,ťovo,ťovu,ťovy,ťově,
ťovou,ťovým,ťových,ťovými,ťovýma

mz3o ť,tě,ti,ťovi,těm,ťů,ťům,tích,ťové,ťův,ťova,ťovo,ťovu,
ťovy,ťově,ťovou,ťovým,ťových,ťovými,ťovýma

mz4 ň,ně,ni,ňovi,něm,ňů,ňům,ních,ňův,ňova,ňovo,ňovu,ňovy,ňově,
ňovou,ňovým,ňových,ňovými,ňovýma

mz4o ň,ně,ni,ňovi,něm,ňů,ňům,ních,ňové,ňův,ňova,ňovo,ňovu,
ňovy,ňově,ňovou,ňovým,ňových,ňovými,ňovýma

mz4e eň,ně,ni,ňovi,něm,ňů,ňům,ních,ňové,ňův,ňova,ňovo,ňovu,
ňovy,ňově,ňovou,ňovým,ňových,ňovými,ňovýma

mz5 ec,ce,ci,covi,cem,če,ců,cům,cích,cův,cova,covo,covu,covy,
cově,covou,covým,cových,covými,covýma,kyně,kyni,kyní,kyň,
kyním,kyních,kyněmi,kynin,kynina,kynino,kyninu,kyniny,kynině,
kyninou,kyniným,kynini,kyniných,kyninými,kyninýma

mz1z 0,e,i,ovi,em,ů,ům,ích,ův,ova,ovo,ovu,ovy,ově,ovou,ovým,
ových,ovými,ovýma,ka,ky,ce,ku,kou,ko,ek,kám,kách,kami,čin,
čina,čino,činu,činy,čině,činou,činým,čini,činých,činými, činýma

mz1zo 0,e,i,ovi,em,ů,ům,ích,ové,ův,ova,ovo,ovu,ovy,ově,ovou,
ovým,ových,ovými,ovýma,ka,ky,ce,ku,kou,ko,ek,kám,kách,kami,
čin,čina,čino,činu,činy,čině,činou,činým,čini,činých,činými,
činýma

mz5e něc,ňce,ňci,ňcovi,ňcem,ňče,ňců,ňcům,ňcích,ňcův,ňcova,
ňcovo,ňcovu,ňcovy,ňcově,ňcovou,ňcovým,ňcových,ňcovými,ňcovýma

mz5i ec,ce,ci,covi,cem,če,ců,cům,cích,cův,cova,covo,covu,covy,
cově,covou,covým,cových,covými,covýma

mz7 0,e,i,ovi,em,ův,ova,ovo,ovu,ovy,ově,ovou,ovým,ových,ovými,
ovýma,kyně,kyni,kyní,kyň,kyním,kyních,kyněmi,kynin,kynina,
kynino,kyninu,kyniny,kynině,kyninou,kyniným,kynini,kyniných,
kyninými,kyninýma

mzs1 0,e,i,ovi,em,ův,ova,ovo,ovu,ovy,ově,ovou,ovým,ových,
ovými,ovýma

sc1 ce,ci,covi,cem,ců,cům,cích,cův,cova,covo,covu,covy,cově,

```
  covou,covým,cových,covými,covýma,kyně,kyni,kyní,kyň,kyním,
  kyních,kyněmi,kynin,kynina,kynino,kyninu,kyniny,kynině,
  kyninou,kyniným,kynini,kyniných,kyninými,kyninýma
jnm í,ího,ímu,ím,ích,ími
jnmp í,ích,ím,ími
hd1 0,u,em,y,ů,ům,ech
hd2 0,u,e,em,y,ů,ům,ech
hd22 0,u,e,em,y,ů,ům,ech
hd4 0,a,u,em,y,ů,ům,ech
hd41 0,a,u,em,y,ů,ům,ech
hd5 0,a,u,e,em,y,ů,ům,ech
hd53 0,a,u,e,em,y,ů,ům,ech
hd54 0,a,u,e,em,y,ů,ům,ech
hd55 0,a,u,e,em,y,ů,ům,ech
hd2x 0,u,ě,em,y,ů,ům,ech
hd2x2 0,u,ě,em,y,ů,ům,ech
hd5x 0,a,u,ě,em,y,ů,ům,ech
hd5x3 0,a,u,ě,em,y,ů,ům,ech
hd5x4 0,a,u,ě,em,y,ů,ům,ech
hd5x5 0,a,u,ě,em,y,ů,ům,ech
hd1xx 0,u,em,y,ů,ům,ech,ích
hd5xx 0,a,u,e,em,y,ů,ům,ech,ích
hd1em em,mu,mem,my,mů,mům,mech
hd1et et,tu,tem,ty,tů,tům,tech
hd2etx et,tu,tě,tem,ty,tů,tům,tech
hd1h h,hu,hem,hy,hů,hům,zích,hách
hd1g g,gu,gem,gy,gů,gům,zích,gách
hd1ch ch,chu,chem,chy,chů,chům,ších,chách
hd1k k,ku,kem,ky,ků,kům,kách,cích
hd2k k,ku,ce,kem,ky,ků,kům,kách,cích
hd2k2 k,ku,ce,kem,ky,ků,kům,kách,cích
hd4k k,ka,ku,kem,ky,ků,kům,kách,cích
hd4k1 k,ka,ku,kem,ky,ků,kům,kách,cích
```

```
hd5k k,ka,ku,ce,kem,ky,ků,kům,kách,cích
hd5k3 k,ka,ku,ce,kem,ky,ků,kům,kách,cích
hd5k4 k,ka,ku,ce,kem,ky,ků,kům,kách,cích
hd5k5 k,ka,ku,ce,kem,ky,ků,kům,kách,cích
hd1ek ek,ku,kem,ky,ků,kům,kách,cích
hd2ek ek,ku,ce,kem,ky,ků,kům,kách,cích
hd2ek2 ek,ku,ce,kem,ky,ků,kům,kách,cích
hd4ek ek,ka,ku,kem,ky,ků,kům,kách,cích
hd4ek1 ek,ka,ku,kem,ky,ků,kům,kách,cích
hd5ek ek,ka,ku,ce,kem,ky,ků,kům,kách,cích
hd5ek3 ek,ka,ku,ce,kem,ky,ků,kům,kách,cích
hd5ek4 ek,ka,ku,ce,kem,ky,ků,kům,kách,cích
hd5ek5 ek,ka,ku,ce,kem,ky,ků,kům,kách,cích
hd1xk něk,ňku,ňkem,ňky,ňků,ňkům,ňkách,ňcích
hd2r r,ru,ře,rem,ry,rů,rům,rech
hd2r2 r,ru,ře,rem,ry,rů,rům,rech
hd5r r,ra,ru,ře,rem,ry,rů,rům,rech
hd5r3 r,ra,ru,ře,rem,ry,rů,rům,rech
hd5r4 r,ra,ru,ře,rem,ry,rů,rům,rech
hd5r5 r,ra,ru,ře,rem,ry,rů,rům,rech
hdus us,u,em,y,ů,ům,ech
hdius us,a,u,em,ů,ům,e,ích,i
hdos os,u,em,y,ů,ům,ech
hdosa os,a,u,em,y,ů,ům,ech
hdes es,u,em,y,ů,ům,ech
hdvus us,u,em,y,ů,ům,ách
hdvos os,u,em,y,ů,ům,ách
hdves es,u,em,y,ů,ům,ách
hdpy y,ů,ům,ech
hdphy y,ů,ům,ách
hdpky ky,ků,kům,kách,cích
mdi1 ý,ého,ému,ém,ým,é,ých,ými
```

```
mdi1s ý,ého,ému,ém,ým
mdi1p é,ých,ým,ými
sj1 0,e,i,em,ů,ům,ích
sj1d ď,dě,di,děm,ďů,ďům,dích
sj1t ť,tě,ti,těm,ťů,ťům,tích
sj1n ň,ně,ni,něm,ňů,ňům,ních
sj1ec ec,ce,ci,cem,ců,cům,cích
sj1en eň,ně,ni,něm,ňů,ňům,ních
sj2 0,e,i,em,u,y,ů,ům,ech
sjp1 e,ů,ům,ích,i
sjp3 e,ů,ům,ech,i
zn1 a,y,u,ou,o,ám,ách,ami,ě,0
zn2e ba,by,bu,bou,bo,bám,bách,bami,bě,eb
zn3 ga,gy,gu,gou,go,gám,gách,gami,ze,g
zn4 ha,hy,hu,hou,ho,hám,hách,hami,ze,h
zn5 cha,chy,chu,chou,cho,chám,chách,chami,še,ch
zn6 ka,ky,ku,kou,ko,kám,kách,kami,ce,k
zn6e ka,ky,ku,kou,ko,kám,kách,kami,ce,ek
zn7 a,y,u,ou,o,ám,ách,ami,e,0
zn7e la,ly,lu,lou,lo,lám,lách,lami,le,el
zn8e ma,my,mu,mou,mo,mám,mách,mami,mě,em
zn9e na,ny,nu,nou,no,nám,nách,nami,ně,en
zn10e ďka,ďky,ďku,ďkou,ďko,ďkám,ďkách,ďkami,ďce,děk
zn11e ťka,ťky,ťku,ťkou,ťko,ťkám,ťkách,ťkami,ťce,těk
zn12e ňka,ňky,ňku,ňkou,ňko,ňkám,ňkách,ňkami,ňce,něk
zn13 ra,ry,ru,rou,ro,rám,rách,rami,ře,r
zn13e ra,ry,ru,rou,ro,rám,rách,rami,ře,er
zn14e ta,ty,tu,tou,to,tám,tách,tami,tě,et
zn15e va,vy,vu,vou,vo,vám,vách,vami,vě,ev
zn16 a,i,u,ou,o,e,ím,ích,emi,í
zn17 ia,ie,i,iu,iou,io,í,iám,iách,iami
zn18 a,y,u,ou,o,ám,ách,ami,i,í
```

```
zn19 a,y,u,ou,o,ám,ách,ami,ji,je,jí,jím,jích,jemi
zn20 a,i,u,ou,o,ám,ách,ami,e,0
zn21 ňa,ni,ňu,ňou,ňo,ňám,ňách,ňami,ně,ň
zn22 ďa,di,ďu,ďou,ďo,ďám,ďách,ďami,dě,ď
zn23 ťa,ti,ťu,ťou,ťo,ťám,ťách,ťami,tě,ť
zn21e ňa,ni,ňu,ňou,ňo,ňám,ňách,ňami,ně,eň
zn25 s,dy,du,dou,do,d,dám,dách,dami
zn26 y,ám,ách,ami,0
zn26a a,0,ám,ách,ami
zn26ak ka,ek,kám,kách,kami
zn26al la,el,lám,lech,lami
zn26an na,en,nám,nech,nami
znp6ge ky,kám,kách,kami,ek
znp10e ďky,ďkám,ďkách,ďkami,děk
znp11e ťky,ťkám,ťkách,ťkami,těk
znp12e ňky,ňkám,ňkách,ňkami,něk
znp13e ry,rám,rách,rami,er
znn 0
zn1n a,y,u,ou,o,ám,ách,ami,ě,0,in,ina,ino,inu,iny,ině,inou,
  iným,ini,iných,inými,inýma
zn6n ka,ky,ku,kou,ko,kám,kách,kami,ce,k,čin,čina,čino,činu,
  činy,čině,činou,činým,čini,činých,činými,činýma
zn6en ka,ky,ku,kou,ko,kám,kách,kami,ce,ek,čin,čina,čino,činu,
  činy,čině,činou,činým,čini,činých,činými,činýma
zn13n ra,ry,ru,rou,ro,rám,rách,rami,ře,r,řin,řina,řino,řinu,
  řiny,řině,řinou,řiným,řini,řiných,řinými,řinýma
zn13en ra,ry,ru,rou,ro,rám,rách,rami,ře,er,řin,řina,řino,řinu,
  řiny,řině,řinou,řiným,řini,řiných,řinými,řinýma
ns3n ně,ni,ní,ním,ních,němi,ň,nin,nina,nino,ninu,niny,nině,
  ninou,niným,nini,ních,ninými,ninýma
ns10n ce,ci,cí,cím,cích,cemi,c,čin,čina,čino,činu,činy,čině,
  činou,činým,čini,činých,činými,činýma
ns1 e,i,í,ím,ích,emi
```

```
ns2 ě,i,í,ím,ích,ěmi
ns10 e,i,í,ím,ích,emi,0
ns3 ně,ni,ní,ním,ních,němi,ň
ns6 e,ím,ích,emi,0
ns7 e,ím,ích,emi,í
ns8 ě,ím,ích,ěmi,í
ps1e eň,ně,ni,ní,ním,ních,němi
ps1 ň,ní,ně,ni,ním,ních,němi
ps2 ev,ve,vi,ví,vím,vích,vemi
ps5 0,e,i,í,ím,ích,emi
ps6 ď,di,dí,dím,dích,děmi,dě
ps7 ť,ti,tí,tím,tích,těmi,tě
ps14 0,i,í,ím,ích,mi
ps18az 0,i,í,ím,ích,emi
ps18 0,i,í,ím,ích,emi,em,ech
ps23 ť,ti,tí,tím,tích,tem,tech,ťmi,tmi
kt1 0,i,í,em,ech,mi
ps24 ť,ti,tí,tě,tím,tích,těmi
ps25 ď,di,dí,dím,dích,dmi
mdf á,é,ou,ých,ým,ými
jnf í,ích,ím,ími
in 0,a,o,u,y,ě,ou,ým,i,ých,ými,ýma
uv ův,ova,ovo,ovu,ovy,ově,ovou,ovým,ovi,ových,ovými,ovýma
mt1 o,a,u,em,0,ům,ech,y
mt4 vo,va,vu,vem,ev,vům,vech,vy
mt1m o,a,u,em,0,ům,ech,i
mt5 bo,ba,bu,bem,eb,bům,bech,by
mt6i o,a,u,em,í,ím,ích,i
mt6 o,a,u,em,í,ům,ech,y
mt7 o,a,u,em,0,ům,ách,y
mt7e ko,ka,ku,kem,ek,kům,kách,ky
mt1x o,a,u,em,0,ům,ech,y,ě
```

```
mt1e o,a,u,em,0,ům,ech,y,e
mt8e lo,la,lu,lem,el,lům,lech,ly,le
mt9e no,na,nu,nem,en,nům,nech,ny,ně
mt7y ko,ka,ku,kem,ek,kům,cích,kách,ky
mt7y0 ko,ka,ku,kem,ek,k,kům,cích,kách,ky
mt3e ro,ra,ru,rem,rům,rech,ry,er
mts o,a,u,em
mt11 o,a,e,u,i,em,0,ům,ech,y
mt12 um,a,u,em,0,ům,ech,y
mt12e rum,ra,ru,rem,rům,rech,ry,er
mt12i um,a,u,em,í,ím,ích,i
mt13 on,a,u,em,0,ům,ech,y
mt13i on,a,u,em,í,ím,ích,i
mt14 mo,ma,mu,mem,em,mům,mech,my
mtp a,0,ům,ech,y
mtp7 ka,ek,kům,kách,ky
mtp8 la,el,lům,lech,ly
mtp9 na,en,nům,nech,ny
mtp3 ra,rům,rech,ry,er
mr1 e,i,em,í,ím,ích
mr10 tě,ti,těm,ť,tím,tích
kr1 e,ete,eti,etem,ata,at,atům,atech,aty
kr1x ě,ěte,ěti,ětem,ata,at,atům,atech,aty
kr2 dě,děte,děti,dětem,ďata,ďat,ďatům,ďatech,ďaty
kr3 tě,těte,těti,tětem,ťata,ťat,ťatům,ťatech,ťaty
kr4 ně,něte,něti,nětem,ňata,ňat,ňatům,ňatech,ňaty
kr5 a,atu,atem,ata,at,atům,atech,aty
st 0,m,ch,mi
mdn é,ého,ému,ém,ým,á,ých,ými
jnn í,ího,ímu,ím,ích,ími
0 0
```

Appendix B

Noun Classification Data

```
>CHARACTER SETS.
{*********************************************************}
{* Used in endings instead of exhaustive listings,       *}
{* in square brackets                                    *}
{*********************************************************}
v       :a,á,e,é,ě,i,í,o,ó,u,ú,ů,y,ý.
sl      :a,á,e,é,ě,i,í,o,ó,u,ú,ů,y,ý,r,l.
t       :h,k,g,r,d,t,n.
vel     :h,k,g.
>ASSIGNMENTS.
{*********************************************************}
{* Anim. masc., Gsg :a                                   *}
{*********************************************************}
>>aa.
pn1     :0,b,l,m,p,s,v,z,d,t,n.
pn2j    :0,át,az,ras,ant,ent,et,ot,ed,jt,kt,lt,mt.
pn3     :0,al,il,el,ěl,f.
pn3j    :0,graf,fob,nom.
pn4j    :0,španěl,manžel,anděl.
pn5     :0,at,it,sed.
pn5j    :0,an.
pn6     :1,r.
pn6f    :1,[v]r.
pn6fy   :1,mistr,nistr.
pn7     :2,es.
```

```
pn8    :2,pes,ves.
pn9    :2,lev.
pn12   :2,ius.
pn13   :2,os.
pn16ik :2,ík.
pn16   :1,ák,k.
pn17   :2,ek.
pn18   :1,ik,ak.
pn19   :1,g.
pn20   :1,vrah.
pn21   :1,h.
pn22   :2,ch.
pn24   :2,ech.
pn26   :0,člověk.
pn27   :3,něk.
pno    :1,o.
pnok   :2,ko.
pnoh   :2,ho.
pnog   :2,go.
pnoc   :1,žo,šo,čo,řo,co,jo,io.
pne    :1,e.
pnus   :2,us.
{*********************************************************}
{* Anim. masc., Gsg :y                                   *}
{*********************************************************}
>>ay.
pd1    :1,[v]a,ba,ca,da,fa,la,ma,na,pa,ra,sa,va,xa,za,ta.
pd2f   :1,[v]sta.
pd2    :1,ita.
pd3    :1,eta,ota,óta.
pd4    :1,ža,ša,ča,řa,ja.
pd4d   :2,ďa.
pd4t   :2,ťa.
pd4n   :2,ňa.
pd5    :2,ka.
pd6    :2,ga.
pd7    :2,ha.
pd8    :3,cha.
{*********************************************************}
```

```
{* Anim. masc., adj. decl. masc. only                    *}
{*******************************************************}
>>a.
mdma   :1,bý,dý,lý,mý,ný,pý,sý,tý,vý,zý.
mdmah  :2,hý.
mdmak  :2,ký.
mdmack :3,cký.
mdmask :3,ský.
mdmach :3,chý.
mdmar  :2,rý.
jnm    :1,í.
{*****************************************}
{* Anim. adj. decl. masc.+fem.             *}
{*****************************************}
>>af.
mdaf   :1,bý,dý,lý,mý,ný,pý,sý,tý,vý,zý.
mdafh  :2,hý.
mdafk  :2,ký.
mdafck :3,cký.
mdafsk :3,ský.
mdafch :3,chý.
mdafr  :2,rý.
jnm    :1,í.
{*******************************************************}
{* Anim. masc., Gsg :e                                 *}
{*******************************************************}
>>ae.
mz1    :0,z,ž,š,král,j.
mz1i   :0,č,ř,c.
mz1e   :0,tel.
mz2    :1,ď.
mz3    :1,ť.
mz4    :1,ň.
mz4e   :2,vězeň,sršeň,eň.
mz5    :2,ec.
mz5i   :2,qqqec.
mz1z   :0,ař,ář,íř,ýř,éř,oř,ač,áč,ič,eč,ěč.
mz1zo  :0,áš,oš.
mz5e   :3,něc.
```

```
mz7    :0,přítel.
sc1    :2,ce.
{*******************************************************}
{* Anim. sing., Gsg :e, adj. decl. masc./masc.+fem.     *}
{*******************************************************}
>>aes.
mzs1   :0,kněz.
mz7    :0,přítel.
{*******************************************************}
{* Anim. plur., Gsg :e, adj. decl. masc./masc.+fem.     *}
{*******************************************************}
>>aep.
jnmp   :1,kněží.
{*******************************************************}
{* Inanim. masc., Gsg :u, Lsg :u, adj. decl.            *}
{*******************************************************}
>>iuu.
hd1    :0,r,d,t,n,b,f,l,m,p,s,x,v,z,c.
hd1xx  :0,botel,motel,hotel,hertz.
hd1em  :2,nájem,zájem.
hd1et  :2,cet,čet.
hd1h   :1,h.
hd1g   :1,g.
hd1ch  :2,ch.
hd1k   :1,vlek,k.
hd1ek  :2,ek.
hd1xk  :3,něk.
{*******************************************************}
{* Inanim. masc., adj. decl.                            *}
{*******************************************************}
>>i.
mdi1   :1,ý.
mdmi   :1,bý,dý,lý,mý,ný,pý,sý,tý,vý,zý.
mdmih  :2,hý.
mdmik  :2,ký.
mdmick :3,cký.
mdmisk :3,ský.
mdmich :3,chý.
mdmir  :2,rý.
```

```
{*********************************************************}
{* Inanim. masc., Gsg :u, Lsg :u, adj. decl. foreign orig.*}
{*********************************************************}
>>iuuc.
hdus   :2,us.
hdos   :2,os.
hdes   :2,es.
hdvus  :2,[vel]us.
hdvos  :2,[vel]os.
hdves  :2,[vel]es.
{*********************************************************}
{* Inanim. masc., sing., adj. decl.                      *}
{*********************************************************}
>>is.
mdi1s  :1,ý.
{*********************************************************}
{* Inanim. masc., plur., adj. decl.                      *}
{*********************************************************}
>>ip.
sjp1   :1,e.
hdpy   :1,y.
hdphy  :1,hy,gy,chy.
hdpky  :2,ky.
mdi1p  :1,é.
{*********************************************************}
{* Inanim. masc., plur., adj. decl.                      *}
{*********************************************************}
>>ieep.
sjp3   :1,e.
{*********************************************************}
{* Inanim. masc., Gsg :a, Lsg :u,                        *}
{*********************************************************}
>>iau.
hd4    :0,r,d,t,n,b,f,l,m,p,s,v,z,c.
hd4k   :1,k.
hd4ek  :2,ek.
hdius  :2,ius.
hdosa  :2,os.
{*********************************************************}
```

```
{* Inanim. masc., Gsg :-u/-a, Lsg :u,                    *}
{*******************************************************}
>>iuau.
hd41   :0,r,d,t,n,b,f,l,m,p,s,v,z,c.
hd4k1  :1,k.
hd4ek1 :2,ek.
{*******************************************************}
{* Inanim. masc., Gsg :u, Lsg :e(ě)                     *}
{*******************************************************}
>>iue.
hd2    :0,l,s,z,c.
hd2x   :0,d,t,n,b,f,m,p,v.
hd2k   :1,k.
hd2ek  :2,ek.
hd2r   :1,r.
{*******************************************************}
{* Inanim. masc., Gsg :u, Lsg :-u/-e(ě)                 *}
{*******************************************************}
>>iuue.
hd22   :0,l,s,z,c.
hd2x2  :0,d,t,n,b,f,m,p,v.
hd2etx :2,účet.
hd2k2  :1,k.
hd2ek2 :2,ek.
hd2r2  :1,r.
{*******************************************************}
{* Inanim. masc., Gsg :a, Lsg :-e(ě)                    *}
{*******************************************************}
>>iae.
hd5    :0,l,s,z.
hd5x   :0,d,t,n,b,f,m,p,v.
hd5xx  :0,les.
hd5k   :1,k.
hd5ek  :2,ek.
hd5r   :1,r.
{*******************************************************}
{* Inanim. masc., Gsg :a, Lsg :u/-e(ě)                  *}
{*******************************************************}
>>iaue.
```

```
hd53   :0,l,s,z.
hd5x3  :0,d,t,n,b,f,m,p,v.
hd5k3  :1,k.
hd5ek3 :2,ek.
hd5r3  :1,r.
{*********************************************************}
{* Inanim. masc., Gsg :-u/-a, Lsg :-e(ě)                 *}
{*********************************************************}
>>iuae.
hd54   :0,l,s,z.
hd5x4  :0,d,t,n,b,f,m,p,v.
hd5k4  :1,k.
hd5ek4 :2,ek.
hd5r4  :1,r.
{*********************************************************}
{* Inanim. masc., Gsg :-u/-a, Lsg :u/-e(ě)               *}
{*********************************************************}
>>iuaue.
hd55   :0,l,s,z.
hd5x5  :0,d,t,n,b,f,m,p,v.
hd5k5  :1,k.
hd5ek5 :2,ek.
hd5r5  :1,r.
{*********************************************************}
{* Inanim. masc., Gsg :e                                 *}
{*********************************************************}
>>ie.
sj1    :0,ž,š,č,ř,c,j,l.
sj1d   :1,ď.
sj1t   :1,ť.
sj1n   :1,ň.
sj1ec  :2,ec.
sj1en  :2,eň.
sj2    :0,en.
{*********************************************************}
{* Feminine                                              *}
{*********************************************************}
>>f.
zn1    :1,da,fa,mba,ta,mpa,vlna,skvrna,srna.
```

```
zn1     :1,[v]na,[v]ma,[v]ba,[v]va,cpa,[sl]pa,[v]ppa.
zn1     :1,[v]mma,[v]bba,[v]tta,[v]vva,[v]nna,[v]dda.
zn2e    :2,ba.
zn3     :2,ga.
zn4     :2,ha.
zn5     :3,cha.
zn6     :2,mlka,[v]ka,[v]kka.
zn6e    :2,ka.
zn7     :1,sa,za,[v]la,[v]lla,[v]xa.
zn7e    :2,la.
zn8e    :2,ma.
zn9e    :2,na.
zn10e   :3,ďka.
zn11e   :3,ťka.
zn12e   :3,ňka.
zn13    :2,[sl]ra.
zn13e   :2,ra.
zn14e   :2,chta.
zn15e   :2,va.
zn16    :1,ja,ia.
zn17    :2,[v]ia.
zn18    :1,oa,ua.
zn19    :1,ea.
zn20    :1,ža,ša,ča,řa,ca.
zn21    :2,[v]ňa.
zn22    :2,ďa.
zn23    :2,ťa.
zn21e   :2,ňa.
zn25    :1,is.
ns1     :1,že,še,če,ře,se,ze,ce,je,ie,le,oe,xe,ve,ue,ye.
ns2     :1,dě,tě,ně,bě,pě,mě,vě.
ns10    :1,ice.
ns3     :2,yně.
ps1e    :2,eň.
ps1     :1,oň,uň,ať,áň,ůň,yň,ýň,iň,íň,óň,úň,lň,rň.
ps2     :2,ev.
ps5     :0,el,[v]j,ýž,ež,áž,oř,ž,š,č,ř,z,c,s,x,l.
ps18az  :0,az.
ps6     :1,ď.
```

```
ps7    :1,ť.
ps25   :1,pověď.
ps18   :0,moc.
ps23   :1,paměť.
ps24   :1,eť.
ps14   :0,p.
kt1    :0,st.
mdf    :1,á.
jnf    :1,í.
znn    :0,i,y.
{*********************************************************}
{* Feminine plur.                                        *}
{*********************************************************}
>>fp.
zn26   :1,y,[v]ky,[v]ry.
znp6ge :2,ky.
znp10e :3,ďky.
znp11e :3,ťky.
znp12e :3,ňky.
znp13e :2,ry.
ns6    :1,ice.
ns7    :1,ce,ve,ie,je,le,ře,se,še,xe,ze,že,ace.
ns8    :1,ě.
zn26a  :1,a,[v]ka,[v]la,[v]na.
zn26ak :2,ka.
zn26al :2,la.
zn26an :2,na.
{*********************************************************}
{* Feminine incl. posses.                                *}
{*********************************************************}
>>fn.
zn1n   :1,da,fa,mba,ta,mpa,vlna,skvrna,srna.
zn1n   :1,[v]na,[v]ma,[v]ba,[v]va,[sl]pa,[v]ppa.
zn1n   :1,[v]mma,[v]bba,[v]tta,[v]vva,[v]nna,[v]dda.
zn6n   :2,mlka,[v]ka,[v]kka.
zn6en  :2,ka.
zn13n  :2,[sl]ra.
zn13en :2,ra.
ns3n   :2,ně.
```

```
ns10n  :2,ice.
{*********************************************************}
{* Neuter                                                *}
{*********************************************************}
>>n.
mt1    :1,[v]no,[sl]mo,[v]vo,[v]co,mbo,nbo,fo,so,zo,[v]ro.
mt1    :1,to,[v]po,do,[sl]bo.
mt4    :2,vo.
mt1m   :1,žo,šo,čo,řo,jo.
mt5    :2,bo.
mt6i   :1,io,yo.
mt6    :1,eo,uo,ao.
mt7    :1,go,[v]ko,ho,cho.
mt1x   :1,sto.
mt7e   :2,ko.
mt1e   :1,[v]lo.
mt8e   :2,lo.
mt9e   :2,no.
mt7y   :2,ísko.
mt7y0  :2,isko.
mt3e   :2,ro.
mt11   :1,meno.
mt12   :2,um,[v]rum,[v]rrum.
mt12e  :3,rum.
mt12i  :2,[v]um.
mt13   :2,on.
mt13i  :2,[v]on.
mt14   :2,mo.
0      :0,á,é,ó,ú,ů,i,y,u.
mr1    :1,ce.
mr10   :2,ště.
kr1    :1,že,še,če,ře,le,se,je.
kr1x   :1,bě,pě,mě,vě.
kr2    :2,dě.
kr3    :2,tě.
kr4    :2,ně.
kr5    :1,ma.
st     :0,í.
{*********************************************************}
```

```
{* Neuter sing.                                                 *}
{************************************************************}
>>ns.
mts    :1,o.
{************************************************************}
{* Neuter plur.                                                 *}
{************************************************************}
>>np.
mtp    :1,[v]ra,[v]ka,[v]la,[v]na,a.
mtp7   :2,ka.
mtp8   :2,la.
mtp9   :2,na.
mtp3   :2,ra.
{************************************************************}
{* Neuter adj.                                                  *}
{************************************************************}
>>nadj.
mdn    :1,é.
jnn    :1,í.
mds    :1,bé,dé,lé,mé,né,pé,sé,té,vé,zé.
mdsh   :2,hé.
mdsk   :2,ké.
mdsck  :3,cké.
mdssk  :3,ské.
mdsch  :3,ché.
mdsr   :2,ré.
{************************************************************}
{* Names masc. anim.                                            *}
{************************************************************}
>>ja.
mz1    :0,z,s,x,ž,š,č,ř,j,c,tel,král.
mz2o   :1,ď.
mz3o   :1,ť.
mz4o   :1,ň.
mz5    :2,ec.
mz5e   :3,něc.
sc1    :2,ce.
pn1    :0,b,l,m,p,v,d,t,n.
pn2    :0,át,ant,ent,et,ot,ed.
```

```
pn3j   :0,al,il,el,ěl,f.
pn4j   :0,španěl,manžel,anděl.
pn5j   :0,at,it,an,sed.
pn6    :1,r.
pn6r   :1,[v]r.
pn7    :2,es.
pn8    :2,pes,ves.
pn9    :2,lev.
pn11   :2,vel,rel,datel.
pn12   :2,ius.
pn13   :2,os.
pn16j  :1,ík,ák,k.
pn17j  :2,ek.
pn18   :1,ik,ak.
pn19   :1,g.
pn20   :1,vrah.
pn21   :1,h.
pn23   :2,ch.
pn24   :2,ech.
pn26   :0,člověk.
pn27j  :3,něk.
pn28j  :3,děk.
pn29j  :3,těk.
pno    :1,o.
pnok   :2,ko.
pnoh   :2,ho.
pnog   :2,go.
pnoc   :1,žo,šo,čo,řo,co,jo,io.
pne    :1,e.
pnus   :2,us.
pd1    :1,ba,ca,da,fa,la,ma,na,pa,ra,sa,va,xa,za,ta.
pd2f   :1,[v]sta.
pd2    :1,ita.
pd3    :1,eta,ota,óta.
pd4    :1,ža,ša,ča,řa,ja.
pd4d   :2,ďa.
pd4t   :2,ťa.
pd4n   :2,ňa.
pd5    :2,ka.
```

```
pd6    :2,ga.
pd7    :2,ha.
pd8    :3,cha.
mdma   :1,bý,dý,lý,mý,ný,pý,sý,tý,vý,zý.
mdmah  :2,hý.
mdmak  :2,ký.
mdmack :3,cký.
mdmask :3,ský.
mdmach :3,chý.
mdmar  :2,rý.
jnm    :1,í.
{*********************************************************}
{* Names masc. anim.  nations                            *}
{*********************************************************}
>>na.
mz5    :2,ec.
mz1z   :0,francouz.
mz5e   :3,něc.
pn3j   :0,al,il,el,ěl,f,b,m,p,s,v,z,n,t,d,r,ch.
pn4j   :0,španěl.
pn4j   :0,at,it,an.
pn6f   :1,[v]r.
pn16   :1,ík,ák,ik,ak.
pn17j  :2,ek.
pn24   :2,ech.
{*********************************************************}
{* Possesives - pure (for irreg. masc. anim., fem. anim.) *}
{*********************************************************}
>>p.
in     :0,in.
uv     :2,ův.
```

Appendix C

Adjective, Adverb and Verb Classes–sets of endings

```
y
    ý,ého,ému,ém,ým,í,á,é,ou,ých,ými,ýma,
    +ější,+ějšího,+ějšímu,+ějším,+ějšími,+ějšíma,+ějších,
    ě,+ěji
ye
    ý,ého,ému,ém,ým,í,á,é,ou,ých,ými,ýma,
    +ejší,+ejšího,+ejšímu,+ejším,+ejšími,+ejšíma,+ejších,
    e,+eji
cky
    cký,ckého,ckému,ckém,ckým,čtí,cká,cké,ckou,ckých,ckými,ckýma,
    +čtější,+čtějšího,+čtějšímu,+čtějším,+čtějšími,
    +čtějšíma,+čtějších,
    cky,+čtěji
sky
    ský,ského,skému,ském,ským,ští,ská,ské,skou,ských,skými,skýma,
    +štější,+štějšího,+štějšímu,+štějším,+štějšími,
    +štějšíma,+štějších,
    sky,+štěji
yx
    ý,ého,ému,ém,ým,á,é,ou,ých,ými,ýma
jev
    ě,+ěji
```

```
ev
    e,+eji
i
    í,ího,ímu,ím,ími,ích,íma
ih
    í,ího,ímu,ím,ími,ích,íma,ě
iv
    í,ího,ímu,ím,ími,ích,íma,
    +ější,+ějšího,+ějšímu,+ějším,+ějšími,+ějšíma,+ějších,
    ě,+ěji
0n
    0
t1n a,ouc,ouce,oucí,oucího,oucímu,oucím,oucími,oucích,oucíma
t2n e,ouc,ouce,oucí,oucího,oucímu,oucím,oucími,oucích,oucíma
t3n e,íc,íce,ící,ícího,ícímu,ícím,ícími,ících,ícíma
t4n ě,íc,íce,ící,ícího,ícímu,ícím,ícími,ících,ícíma
md 0,ši,še,ší,šího,šímu,ším,šími,ších,šíma
0ns 0
i1 0,me,te
i2 i,eme,ete
i3 i,ěme,ěte
p1 u,eš,e,eme,ete,ou
p7 u,ěš,ě,ěme,ěte,ou
p2 i,u,eš,e,eme,ete,í,ou
p3 ím,íš,í,íme,íte
p4 ím,íš,í,íme,íte,ejí
p5 ím,íš,í,íme,íte,ějí
p6 ám,áš,á,áme,áte,ají
r 0,a,o,i,y
re el,la,lo,li,ly
rx a,o,i,y
s1 0,a,u,o,i,y,í,ím,ích,ími
s2 0,a,u,o,i,y,í,ím,ích,ími,ý,ého,ému,ém,ým,á,é,ou,ých,ými,ýma,
    +ější,+ějšího,+ějšímu,+ějším,+ějšími,+ějších,+ějšíma,
    ě,+ěji
s3 t,n,na,nu,no,ni,ny,ní,ním,ních,ními
s4 t,n,na,nu,no,ni,ny,ní,ním,ních,ními,
```

```
    ný,ného,nému,ném,ným,ná,né,nou,ných,nými,nýma,
    +nější,+nějšího,+nějšímu,+nějším,+nějšími,+nějších,+nějšíma,
    ně,+něji
s5 0,a,u,o,i,y,í,ý,ého,ému,ém,ým,á,é,ou,ých,ými,ýma,
    +ější,+ějšího,+ějšímu,+ějším,+ějšími,+ějších,+ějšíma,
    ě,+ěji
ns í,ím,ích,ími
a í,ý,ého,ému,ém,ým,á,é,ou,ých,ými,ýma,
    +ější,+ějšího,+ějšímu,+ějším,+ějšími,+ějších,+ějšíma,
    ě,+ěji
ae í,ý,ého,ému,ém,ým,á,é,ou,ých,ými,ýma,
    +ejší,+ejšího,+ejšímu,+ejším,+ejšími,+ejších,+ejšíma,
    e,+eji
t t
noutn nout,ni,něme,něte,nu,neš,ne,neme,nete,nou,
    na,nouc,nouce,
    noucí,noucího,noucímu,noucím,noucími,noucích,noucíma,
    l,la,lo,li,ly,
    nut,nuta,nutu,nuto,nuti,nuty,
    nutý,nutá,nuté,nutého,nutému,nutém,nutým,nutou,
    nutí,nutých,nutými,nutýma,
    +nutější,+nutějšího,+nutějšímu,+nutějším,+nutějšími,
    +nutějších,+nutějšíma,
    nutím,nutích,nutími
ovatn ovat,uj,ujme,ujte,uji,uju,uješ,uje,ujeme,ujete,ují,
    ujíc,ujíce,
    ující,ujícího,ujícímu,ujícím,ujícími,ujících,ujícíma,
    ovací,ovacího,ovacímu,ovacím,ovacími,ovacích,ovacíma,
    oval,ovala,ovalo,ovali,ovaly,
    ován,ována,ovánu,ováno,ováni,ovány,
    ovaný,ovaná,ované,ovaného,ovanému,ovaném,ovaným,ovanou,
    ovaní,ovaných,ovanými,ovaným,
    +ovanější,+ovanějšího,+ovanějšímu,+ovanějším,+ovanějšími,
    +ovanějších,+ovanějšíma,
    ovaně,+ovaněji,
    ovávat,ovávej,ovávejme,ovávejte,ovávám,ováváš,ovává,
    ováváme,ováváte,ovávají,
```

```
    ovávaje,ovávajíc,ovávajíce,
    ovávající,
    ovávajícího,ovávajícímu,ovávajícím,ovávajícími,ovávajících,
    ovávajícíma,
    ovávací,
    ovávacího,ovávacímu,ovávacím,ovávacími,ovávacích,ovávacíma,
    ováva1,ováva1a,ováva1o,ováva1i,ováva1y,
    ováván,ovávána,ováván,ováváno,ováváni,ovávány,
    ovávaný,ovávaná,ovávané,
    ovávaného,ovávanému,ovávaném,ovávaným,ovávanou,
    ovávaní,ovávaných,ovávanými,ovávaným
it0n it,0,me,te,ím,íš,í,íme,íte,
    ě,íc,íce,
    ící,ícího,ícímu,ícím,ícími,ících,ícíma,
    il,ila,ilo,ili,ily,
    en,ena,enu,eno,eni,eny,
    ený,ená,ené,ného,nému,ném,ným,enou,
    ení,ených,enými,enýma,
    +ěnější,+ějšího,+ějšímu,+ějším,+ějšími,+ějších,
    +ějšíma,
    ením,ních,ními,
    ně,+ěji,
    ívat,ívám,íváš,ívá,íváme,íváte,ívají,
    ívaje,ívajíc,ívajíce,
    ívající,
    ívajícího,ívajícímu,ívajícím,ívajícími,ívajících,ívajícíma,
    íval,ívala,ívalo,ívali,ívaly,
    íván,ívána,íván,íváno,íváni,ívány,
    ívaný,ívaná,ívané,
    ívaného,ívanému,ívaném,ívaným,ívanou,
    ívaní,ívaných,ívanými,ívaným
itin it,i,ěme,ěte,ím,íš,í,íme,íte,
    ě,íc,íce,
    ící,ícího,ícímu,ícím,ícími,ících,ícíma,
    il,ila,ilo,ili,ily,
    en,ena,enu,eno,eni,eny,
    ený,ená,ené,ného,nému,ném,ným,enou,
```

```
    ení,ených,enými,enýma,
    +enější,+enějšího,+enějšímu,+enějším,+enějšími,+enějších,
    +enějšíma,
    ením,ených,enými,
    ně,+ěji,
    ívat,ívám,íváš,ívá,íváme,íváte,ívají,
    ívaje,ívajíc,ívajíce,
    ívající,
    ívajícího,ívajícímu,ívajícím,ívajícími,ívajících,ívajícíma,
    íval,ívala,ívalo,ívali,ívaly,
    íván,ívána,ívánu,íváno,íváni,ívány,
    ívaný,ívaná,ívané,
    ívaného,ívanému,ívaném,ívaným,ívanou,
    ívaní,ívaných,ívanými,ívanýma
iten it,0,me,te,ím,íš,í,íme,íte,
    e,íc,íce,
    ící,ícího,ícímu,ícím,ícími,ících,ícíma,
    il,ila,ilo,ili,ily,
    en,ena,enu,eno,eni,eny,
    ený,ená,ené,ného,nému,ném,ným,enou,
    ení,ených,enými,enýma,
    +enější,+enějšího,+enějšímu,+enějším,+enějšími,+enějších,
    +enějšíma,
    ením,ených,enými,
    ně,+ěji,
    ívat,ívám,íváš,ívá,íváme,íváte,ívají,
    ívaje,ívajíc,ívajíce,
    ívající,
    ívajícího,ívajícímu,ívajícím,ívajícími,ívajících,ívajícíma,
    íval,ívala,ívalo,ívali,ívaly,
    íván,ívána,ívánu,íváno,íváni,ívány,
    ívaný,ívaná,ívané,
    ívaného,ívanému,ívaném,ívaným,ívanou,
    ívaní,ívaných,ívanými,ívanýma
titn tit,ť,ťme,ťte,tím,tíš,tí,tíme,títe,
    tě,tíc,tíce,
    tící,tícího,tícímu,tícím,tícími,tících,tícíma,
```

```
    til,tila,tilo,tili,tily,
    cen,cena,cenu,ceno,ceni,ceny,
    cený,cená,cené,cenéh o,cenému,ceném,cenýmcenou,
    cení,cených,cenými,cenýma,
    +cenější,+cenějšího,+cenějšímu,+cenějším,+cenějšími,
    +cenějších,+cenějšíma,
    cením,ceních,ceními,
    ceně,+ceněji,
    cenost,ceností,cenostem,cenostech,cenostmi,
    tívat,tívám,tíváš,tívá,tíváme,tíváte,tívají,
    tívaje,tívajíc,tívajíce,
    tívající,
    tívajícího,tívajícímu,tívajícím,tívajícími,tívajících,
    tívajícíma,
    tíval,tívala,tívalo,tívali,tívaly,
    tíván,tívána,tíváno,tíváni,tívány,
    tívaný,tívaná,tívané,
    tívaného,tívanému,tívaném,tívaným,tívanou,
    tívaní,tívaných,tívanými,tívanýma
ditn dit,ď,ďme,ďte,dím,díš,dí,díme,díte,
    dě,díc,díce,
    dící,dícího,dícímu,dícím,dícími,dících,dícíma,
    dil,dila,dilo,dili,dily,
    zen,zena,zenu,zeno,zeni,zeny,
    zený,zená,zené,zeného,zenému,zeném,zeným,zenou,
    zení,zených,zenými,zenýma,
    +zenější,+zenějšího,+zenějšímu,+zenějším,+zenějšími,
    +zenějších,+zenějšíma,
    zením,zeních,zeními,
    zeně,+zeněji,
    zenost,zeností,zenostem,zenostech,zenostmi,
    dívat,dívám,díváš,dívá,díváme,díváte,dívají,
    dívaje,dívajíc,dívajíce,
    dívající,
    dívajícího,dívajícímu,dívajícím,dívajícími,dívajících,
    dívajícíma,
    díval,dívala,dívalo,dívali,dívaly,
```

```
    díván,díván a,díván u,díván o,díváni,dívány,
    dívaný,dívaná,dívané,
    dívaného,dívanému,dívaném,dívaným,dívanou,
    dívaní,dívaných,dívanými,dívaný ma
nitn nit,ň,ňme,ňte,ním,níš,ní,níme,níte,
    ně,níc,níce,
    nící,nícího,nícímu,nícím,nícími,nících,nícíma,
    nil,nila,nilo,nili,nily,
    něn,něna,něnu,něno,něni,něny,
    něný,něná,něné,něného,něnému,něném,něným,něnou,
    není,něných,něnými,něný ma,
    +nější,+nějšího,+nějšímu,+nějším,+nějšími,
    +nějších,+nějšíma,
    něním,něních,něními,
    ně,+nějí,
    nívat,nívám,níváš,nívá,níváme,níváte,nívají,
    nívaje,nívajíc,nívajíce,
    nívající,
    nívajícího,nívajícímu,nívajícím,nívajícími,nívajících,
    nívajícíma,
    níval,nívala,nívalo,nívali,nívaly,
    níván,nívána,níván u,níváno,níváni,nívány,
    nívaný,nívaná,nívané,
    nívaného,nívanému,nívaném,nívaným,nívanou,
    nívaní,nívaných,nívanými,nívaný ma
itxn it,ím,íš,í,íme,íte,
    il,ila,ilo,ili,ily
etn et,ej,ejme,ejte,ím,íš,í,íme,íte,ejí,
    eje,ejíc,ejíce,
    ející,ejícího,ejícímu,ejícím,ejícími,ejících,ejícíma,
    el,ela,elo,eli,ely,
    en,ena,enu,eno,eni,eny,
    ený,ená,ené,ného,nému,ném,ným,enou,
    ení,ených,enými,enýma,
    +nější,+nějšího,+nějšímu,+nějším,+nějšími,+nějších,
    +nějšíma,
    ením,ení ch,ení mi,
```

```
    ně,+něji,
    ívat,ívám,íváš,ívá,íváme,íváte,ívají,
    ívaje,ívajíc,ívajíce,
    ívající,
    ívajícího,ívajícímu,ívajícím,ívajícími,ívajících,ívajícíma,
    íval,ívala,ívalo,ívali,ívaly,
    íván,ívána,ívánu,íváno,íváni,ívány,
    ívaný,ívaná,ívané,
    ívaného,ívanému,ívaném,ívaným,ívanou,
    ívaní,ívaných,ívanými,ívanýma
ětn ět,ěj,ějme,ějte,ím,íš,í,íme,íte,ějí,
    ěje,ějíc,ějíce,
    ějící,ějícího,ějícímu,ějícím,ějícími,ějících,ějícíma,
    ěl,ěla,ělo,ěli,ěly,
    ěn,ěna,ěnu,ěno,ěni,ěny,
    ěný,ěná,ěné,ěného,ěnému,ěném,ěným,ěnou,
    ění,ěných,ěnými,ěnýma,
    +ěnější,+ějšího,+ějšímu,+ějším,+ějšími,+ějších,
    +ějšíma,
    ěním,ěních,ěními,
    ně,+ěji,
    ívat,ívám,íváš,ívá,íváme,íváte,ívají,
    ívaje,ívajíc,ívajíce,
    ívající,
    ívajícího,ívajícímu,ívajícím,ívajícími,ívajících,ívajícíma,
    íval,ívala,ívalo,ívali,ívaly,
    íván,ívána,ívánu,íváno,íváni,ívány,
    ívaný,ívaná,ívané,
    ívaného,ívanému,ívaném,ívaným,ívanou,
    ívaní,ívaných,ívanými,ívanýma
atn at,ej,ejme,ejte,ám,áš,á,áme,áte,ají,
    aje,ajíc,ajíce,
    ající,ajícího,ajícímu,ajícím,ajícími,ajících,ajícíma,
    ací,acího,acímu,acím,acími,acích,acíma,
    al,ala,alo,ali,aly,
    án,ána,ánu,áno,áni,ány,
    aný,aná,ané,ného,anému,aném,aným,anou,
```

```
    aní,aných,anými,anýma,
    +anější,+anějšího,+anějšímu,+anějším,+anějšími,+anějších,
    +anějšíma,
    ání,áním,áních,ními,
    ně,+ěji,
    ávat,ávej,ávejme,ávejte,ávám,áváš,ává,áváme,váte,ávají,
    ávaje,ávajíc,ávajíce,
    ávající,ávajícího,ávajícímu,ávajícím,ávajícími,ávajících,
    ávajícíma,
    ávací,ávacího,ávacímu,ávacím,ávacími,ávacích,ávacíma,
    ával,ávala,ávalo,ávali,ávaly,
    áván,ávána,ávánu,áváno,áváni,ávány,
    ávaný,ávaná,ávané,ávaného,ávanému,ávaném,ávaným,ávanou,
    ávaní,ávaných,ávanými,ávanýma,
    +ávanější,+ávanějšího,+ávanějšímu,+ávanějším,+ávanějšími,
    +ávanějších,+ávanějšíma
noutd nout,ni,něme,něte,nu,neš,ne,neme,nete,nou,
    l,la,lo,li,ly,
    nuv,nuvši,nuvše,
    nuvší,nuvšího,nuvšímu,nuvším,nuvšími,nuvších,nuvšíma,
    nut,nuta,nutu,nuto,nuti,nuty,
    nutý,nutá,nuté,nutého,nutému,nutém,nutým,nutou,
    nutí,nutých,nutými,nutýma,
    +nutější,+nutějšího,+nutějšímu,+nutějším,+nutějšími,
    +nutějších,+nutějšíma
ovatd ovat,uj,ujme,ujte,uji,uju,uješ,uje,ujeme,ujete,ují,
    oval,ovala,ovalo,ovali,ovaly,
    ovav,ovavši,ovavše,
    ovavší,ovavšího,ovavšímu,ovavším,ovavšími,ovavších,ovavšíma,
    ován,ována,ovánu,ováno,ováni,ovány,
    ovaný,ovaná,ované,ovaného,ovanému,ovaném,ovaným,ovanou,
    ovaní,ovaných,ovanými,ovanýma,
    +ovanější,+ovanějšího,+ovanějšímu,+ovanějším,+ovanějšími,
    +ovanějších,+ovanějšíma,
    ovávat,ovávej,ovávejme,ovávejte,ovávám,ováváš,ovává,
    ováváme,ováváte,ovávají,
    ovávaje,ovávajíc,ovávajíce,
```

```
    ovávající,
    ovávajícího,ovávajícímu,ovávajícím,ovávajícími,ovávajících,
    ovávajícíma,
    ovávací,
    ovávacího,ovávacímu,ovávacím,ovávacími,ovávacích,
    ovávacíma,
    ovával,ovávala,ovávalo,ovávali,ovávaly,
    ováván,ovávána,ováváno,ováváni,ovávány,
    ovávaný,ovávaná,ovávané,
    ovávaného,ovávanému,ovávaném,ovávaným,ovávanou,
    ovávaní,ovávaných,ovávanými,ovávaným
it0d it,0,me,te,ím,íš,í,íme,íte,
    il,ila,ilo,ili,ily,
    iv,ivši,ivše,
    ivší,ivšího,ivšímu,ivším,ivšími,ivších,ivšíma,
    en,ena,enu,eno,eni,eny,
    ený,ená,ené,ného,nému,ném,ným,enou,
    ení,ených,enými,enýma,
    +enější,+enějšího,+enějšímu,+enějším,+enějšími,
    +enějších,+enějšíma,
    ením,ení,eními,
    ně,+ěji
itid it,i,ěme,ěte,ím,íš,í,íme,íte,
    il,ila,ilo,ili,ily,
    iv,ivši,ivše,
    ivší,ivšího,ivšímu,ivším,ivšími,ivších,ivšíma,
    en,ena,enu,eno,eni,eny,
    ený,ená,ené,ného,nému,ném,ným,enou,
    ení,ených,enými,enýma,
    +enější,+enějšího,+enějšímu,+enějším,+enějšími,+enějších,
    +enějšíma,
    ením,ení,eními,
    ně,+ěji
ited it,0,me,te,ím,íš,í,íme,íte,
    il,ila,ilo,ili,ily,
    iv,ivši,ivše,
    ivší,ivšího,ivšímu,ivším,ivšími,ivších,ivšíma,
```

```
    en,ena,enu,eno,eni,eny,
    ený,ená,ené,ného,nému,ném,ným,enou,
    ení,ných,nými,ných,ných,nými,ných,ných,
    +nější,+nějšího,+nějšímu,+nějším,+nějšími,+nějších,
    +nějšíma,
    ením,ních,ními,
    ně,+něji
titd tit,ť,ťme,ťte,tím,tíš,tí,tíme,títe,
    til,tila,tilo,tili,tily,
    tiv,tivši,tivše,
    tivší,tivšího,tivšímu,tivším,tivšími,tivších,tivšíma,
    cen,cena,cenu,ceno,ceni,ceny,
    cený,cená,cené,ceného,cenému,ceném,ceným,cenou,
    cení,cených,cenými,cenýma,
    +cenější,+cenějšího,+cenějšímu,+cenějším,+cenějšími,
    +cenějších,+cenějšíma,
    cením,ceních,cními,
    ceně,+ceněji
ditd dit,ď,ďme,ďte,dím,díš,dí,díme,díte,
    dil,dila,dilo,dili,dily,
    div,divši,divše,
    divší,divšího,divšímu,divším,divšími,divších,divšíma,
    zen,zena,zenu,zeno,zeni,zeny,
    zený,zená,zené,zeného,zenému,zeném,zeným,zenou,
    zení,zených,zenými,zenýma,
    +zenější,+zenějšího,+zenějšímu,+zenějším,+zenějšími,
    +zenějších,+zenějšíma,
    zením,zeních,zními,
    zeně,+zeněji
nitd nit,ň,ňme,ňte,ním,níš,ní,níme,níte,
    nil,nila,nilo,nili,nily,
    niv,nivši,nivše,
    nivší,nivšího,nivšímu,nivším,nivšími,nivších,nivšíma,
    něn,něna,něnu,něno,něni,něny,
    něný,něná,něné,něného,něnému,něném,něným,něnou,
    není,něných,něnými,něnýma,
    +něnější,+něnějšího,+něnějšímu,+něnějším,+něnějšími,
```

```
    +něnějších,+něnějšíma,
    něním,něních,něními,
    něně,+něněji
itxd it,ím,íš,í,íme,íte,
    il,ila,ilo,ili,ily,
    iv,ivši,ivše,
    ivší,ivšího,ivšímu,ivším,ivšími,ivších,ivšíma
etd et,ej,ejme,ejte,ím,íš,í,íme,íte,ejí,
    el,ela,elo,eli,ely,
    ev,evši,evše,
    evší,evšího,evšímu,evším,evšími,evších,evšíma,
    en,ena,enu,eno,eni,eny,
    ený,ená,ené,eného,enému,eném,eným,enou,
    ení,ených,enými,enýma,
    +enější,+enějšího,+enějšímu,+enějším,+enějšími,+enějších,
    +enějšíma,
    ením,eních,eními,
    ně,+ěneji
ětd ět,ěj,ějme,ějte,ím,íš,í,íme,íte,ějí,
    ěl,ěla,ělo,ěli,ěly,
    ěv,ěvši,ěvše,
    ěvší,ěvšího,ěvšímu,ěvším,ěvšími,ěvších,ěvšíma,
    ěn,ěna,ěnu,ěno,ěni,ěny,
    ěný,ěná,ěné,ěného,ěnému,ěném,ěným,ěnou,
    ění,ěných,ěnými,ěnýma,
    +ěnější,+ěnějšího,+ěnějšímu,+ěnějším,+ěnějšími,+ěnějších,
    +ěnějšíma,
    ěním,ěních,ěními,
    ěně,+ěněji
atd at,ej,ejme,ejte,ám,áš,á,áme,áte,ají,
    al,ala,alo,ali,aly,
    av,avši,avše,
    avší,avšího,avšímu,avším,avšími,avších,avšíma,
    án,ána,ánu,áno,áni,ány,
    aný,aná,ané,ánho,anému,aném,aným,anou,
    aní,aných,anými,anýma,
    +anější,+anějšího,+anějšímu,+anějším,+anějšími,
```

+anějších,+anějšíma,
ávat,ávej,ávejme,ávejte,ávám,áváš,ává,áváme,
áváte,ávají,
ávaje,ávajíc,ávajíce,
ávající,ávajícího,ávajícímu,ávajícím,ávajícími,
ávajících,ávajícíma,
ávací,ávacího,ávacímu,ávacím,ávacími,ávacích,ávacíma,
ával,ávala,ávalo,ávali,ávaly,
áván,ávána,ávánu,áváno,áváni,ávány,
ávaný,ávaná,ávané,ávaného,ávanému,ávaném,ávaným,ávanou,
ávaní,ávaných,ávanými,ávaným,
+ávanější,+ávanějšího,+ávanějšímu,+ávanějším,+ávanějšími,
+ávanějších,+ávanějšíma

Appendix D

Czech Morphology Rules

Examples

```
{ pattern = tiskne }
<_><n><o><u><t>$ :=
    [key=<_>nout,x=(noutn|noutd),cat=[pos=v,v.type=full],
    morf=[infl=[pf=[v.form=inf]]]];
<_><n><o><u><t><i>$ :=
    [key=<_>nout,x=(noutn|noutd),
    cat=[pos=v,v.type=full],
    morf=[infl=[pf=[v.form=inf]]]];
<_><n><i>$ := [key=<_>nout,x=(noutn|noutd),
    cat=[pos=v,v.type=full],
    morf=[infl=[pf=[v.form=fin,mode=imp],
    raf=[pers=2,num=sg]]]];
<_><n><ě><m><e>$ := [key=<_>nout,x=(noutn|noutd),
    cat=[pos=v,v.type=full],
    morf=[infl=[pf=[v.form=fin,mode=imp],
    raf=[pers=1,num=pl]]]];
<_><n><ě><t><e>$ := [key=<_>nout,x=(noutn|noutd),
    cat=[pos=v,v.type=full],
    morf=[infl=[pf=[v.form=fin,mode=imp],
    raf=[pers=2,num=pl]]]];
<_><n><u>$ := [key=<_>nout,x=(noutn|noutd),
```

```
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=fin,mode=ind,
   tense=pres],
   raf=[pers=1,num=sg]]]];
<_><n><e><š>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=fin,mode=ind,
   tense=pres],
   raf=[pers=2,num=sg]]]];
<_><n><e>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=fin,mode=ind,
   tense=pres],
   raf=[pers=3,num=sg]]]];
<_><n><e><m><e>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=fin,mode=ind,
   tense=pres],
   raf=[pers=1,num=pl]]]];
<_><n><e><t><e>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=fin,mode=ind,
   tense=pres],
   raf=[pers=2,num=pl]]]];
<_><n><o><u>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=fin,mode=ind,
   tense=pres],
   raf=[pers=3,num=pl]]]];
<_><l>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=pastp],
   raf=[gender=(a|i),num=sg]]]];
<_><l><a>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=pastp],
   raf=([gender=f,num=sg]
```

```
   | [gender=n,num=pl])]]];
<_><l><o>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=pastp],
   raf=[gender=n,num=sg]]]];
<_><l><i>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=pastp],
   raf=[gender=a,num=pl]]]];
<_><l><y>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=pastp],
   raf=[gender=(i|f),num=pl]]]];
<_><n><u><t>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=passp],
   raf=[gender=(a|i),num=sg]]]];
<_><n><u><t><a>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=passp],
   raf=([gender=f,num=sg]
   | [gender=n,num=pl])]]];
<_><n><u><t><o>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=passp],
   raf=[gender=n,num=sg]]]];
<_><n><u><t><i>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=passp],
   raf=[gender=a,num=pl]]]];
<_><n><u><t><y>$ := [key=<_>nout,x=(noutn|noutd),
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=passp],
   raf=[gender=(i|f),num=pl]]]];
<_><n><a>$ := [key=<_>nout,x=noutn,
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=transgr,rtense=con],
```

```
   raf=[gender=(a|i),num=sg]]]];
<_><n><o><u><c>$ := [key=<_>nout,x=noutn,
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=transgr,rtense=con],
   raf=[gender=(f|n),num=sg]]]];
<_><n><o><u><c><e>$ := [key=<_>nout,x=noutn,
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=transgr,rtense=con],
   raf=[num=pl]]]];
<_><n><u><v>$ := [key=<_>nout,x=noutd,
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=transgr,rtense=pre],
   raf=[gender=(a|i),num=sg]]]];
<_><n><u><v><š><i>$ := [key=<_>nout,x=noutd,
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=transgr,rtense=pre],
   raf=[gender=(f|n),num=sg]]]];
<_><n><u><v><š><e>$ := [key=<_>nout,x=noutd,
   cat=[pos=v,v.type=full],
   morf=[infl=[pf=[v.form=transgr,rtense=pre],
   raf=[num=pl]]]];


{ adverbs derived from adjectives }
<_><ě>$ := [key=<_>ý,x=y,hw=<_>ě,cat=[pos=ad],
   morf=[infl=[pf=[compar=base]]]];
<_><ě>$ :=
   [key=<_>í,x=(ih|iv),hw=<_>ě,cat=[pos=ad],
   morf=[infl=[pf=[compar=base]]]];
<_><e>$ := [key=<_>ý,x=ye,hw=<_>e,cat=[pos=ad],
   morf=[infl=[pf=[compar=base]]]];
<_><y>$ :=
   [key=<_>ý,x=(cky|sky),hw=<_>y,cat=[pos=ad],
   morf=[infl=[pf=[compar=base]]]];
<_><ě><j><i>$ :=
   [key=<_>ý,x=y,hw=<_>ě,cat=[pos=ad],
   morf=[infl=[pf=[compar=(comp|sup)]]]];
```

```
<_><ě><j><i>$ :=
   [key=<_>í,x=iv,hw=<_>ě,cat=[pos=ad],
   morf=[infl=[pf=[compar=(comp|sup)]]]];
<_><e><j><i>$ :=
   [key=<_>ý,x=ye,hw=<_>e,cat=[pos=ad],
   morf=[infl=[pf=[compar=(comp|sup)]]]];
<_><č><t><ě><j><i>$ :=
   [key=<_>cký,x=cky,hw=<_>cky,cat=[pos=ad],
   morf=[infl=[pf=[compar=(comp|sup)]]]];
<_><š><t><ě><j><i>$ :=
   [key=<_>ský,x=sky,hw=<_>sky,cat=[pos=ad],
   morf=[infl=[pf=[compar=(comp|sup)]]]];


{ pattern = otcův (independent possesives,NOT derivations) }
<_><ů><v>$ := [key=<_>ův,x=uv,cat=[pos=a,a.type=poss],
   morf=[infl=[pf=[gender=a],
   raf=([gender=a,num=sg,case=(nom|voc)]
   | [gender=i,num=sg,case=(nom|acc|voc)])]]];
<_><o><v><a>$ :=
   [key=<_>ův,x=uv,cat=[pos=a,a.type=poss],
   morf=[infl=[pf=[gender=a],
   raf=([gender=a,num=sg,case=(gen|acc)]
   | [gender=(i|n),num=sg,case=gen]
   | [gender=f,num=sg,case=(nom|voc)]
   | [gender=n,num=pl,case=(nom|acc|voc)])]]];
<_><o><v><u>$ :=
   [key=<_>ův,x=uv,cat=[pos=a,a.type=poss],
   morf=[infl=[pf=[gender=a],
   raf=([gender=(a|i|n),num=sg,case=(dat|loc)]
   | [gender=f,num=sg,case=acc])]]];
<_><o><v><ě>$ :=
   [key=<_>ův,x=uv,cat=[pos=a,a.type=poss],
   morf=[infl=[pf=[gender=a],
   raf=([gender=(a|i|n),num=sg,case=loc]
   | [gender=f,num=sg,case=(dat|loc)])]]];
<_><o><v><ý><m>$ :=
```

```
    [key=<_>ův,x=uv,cat=[pos=a,a.type=poss],
    morf=[infl=[pf=[gender=a],
    raf=([gender=(a|i|n),num=sg,case=ins]
    | [num=pl,case=dat])]]];
<_><o><v><y>$ :=
    [key=<_>ův,x=uv,cat=[pos=a,a.type=poss],
    morf=[infl=[pf=[gender=a],
    raf=([gender=f,num=sg,case=gen]
    | [gender=(i|f),num=pl,case=(nom|acc|voc)]
    | [gender=a,num=pl,case=acc])]]];
<_><o><v><o><u>$ :=
    [key=<_>ův,x=uv,cat=[pos=a,a.type=poss],
    morf=[infl=[pf=[gender=a],
    raf=[gender=f,num=sg,case=ins]]]];
<_><o><v><o>$ :=
    [key=<_>ův,x=uv,cat=[pos=a,a.type=poss],
    morf=[infl=[pf=[gender=a],
    raf=[gender=n,num=sg,case=(nom|acc|voc)]]]];
<_><o><v><i>$ :=
    [key=<_>ův,x=uv,cat=[pos=a,a.type=poss],
    morf=[infl=[pf=[gender=a],
    raf=[gender=a,num=pl,case=(nom|voc)]]]];
<_><o><v><ý><c><h>$ :=
    [key=<_>ův,x=uv,cat=[pos=a,a.type=poss],
    morf=[infl=[pf=[gender=a],
    raf=[num=pl,case=(gen|loc)]]]];
<_><o><v><ý><m><i>$ :=
    [key=<_>ův,x=uv,cat=[pos=a,a.type=poss],
    morf=[infl=[pf=[gender=a],
    raf=[num=pl,case=ins]]]];
<_><o><v><ý><m><a>$ :=
    [key=<_>ův,x=uv,cat=[pos=a,a.type=poss],
    morf=[infl=[pf=[gender=a],
    raf=[num=du,case=ins]]]];


{ class = kuře (nouns,neuter,-e,gen sg -ete); all subclasses }
```

```
<_><e>$ := [key=<_>e,x=kr1,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=sg,case=(nom|acc|voc)]]]];
<_><ě>$ := [key=<_>ě,x=(kr1x|kr4),cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=sg,case=(nom|acc|voc)]]]];
<_><a>$ := [key=<_>a,x=kr5,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=sg,case=(nom|acc|voc)]]]];
<_><t><e>$ := [key=<_>,x=(kr1|kr1x|kr4),cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=sg,case=gen]]]];
<_><a><t><u>$ := [key=<_>a,x=kr5,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=sg,case=(gen|dat|loc)]]]];
<_><t><i>$ := [key=<_>,x=(kr1|kr1x|kr4),cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=sg,case=(dat|loc)]]]];
<_><t><e><m>$ :=
   [key=<_>,x=(kr1|kr1x|kr4),cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=sg,case=ins]]]];
<_><a><t><a>$ := [key=<_>e,x=kr1,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=(nom|acc|voc)]]]];
<_><a><t><a>$ := [key=<_>ě,x=kr1x,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=(nom|acc|voc)]]]];
<_><ň><a><t><a>$ := [key=<_>ně,x=kr4,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=(nom|acc|voc)]]]];
<_><a><t><a>$ := [key=<_>a,x=kr5,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=(nom|acc|voc)]]]];
<_><a><t>$ := [key=<_>e,x=kr1,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=gen]]]];
<_><a><t>$ := [key=<_>ě,x=kr1x,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=gen]]]];
<_><ň><a><t>$ := [key=<_>ně,x=kr4,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=gen]]]];
<_><a><t>$ := [key=<_>a,x=kr5,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=gen]]]];
<_><a><t><ů><m>$ := [key=<_>e,x=kr1,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=dat]]]];
<_><a><t><ů><m>$ := [key=<_>ě,x=kr1x,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=dat]]]];
<_><ň><a><t><ů><m>$ :=
   [key=<_>ně,x=kr4,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=dat]]]];
<_><a><t><ů><m>$ := [key=<_>a,x=kr5,cat=[pos=n],
```

```
   morf=[infl=[pf=[gender=n,num=pl,case=dat]]]];
<_><a><t><e><c><h>$ :=
   [key=<_>e,x=kr1,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=loc]]]];
<_><a><t><e><c><h>$ :=
   [key=<_>ě,x=kr1x,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=loc]]]];
<_><ň><a><t><e><c><h>$ :=
   [key=<_>ně,x=kr4,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=loc]]]];
<_><a><t><e><c><h>$ :=
   [key=<_>a,x=kr5,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=loc]]]];
<_><a><t><y>$ := [key=<_>e,x=kr1,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=ins]]]];
<_><a><t><y>$ := [key=<_>ě,x=kr1x,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=ins]]]];
<_><ň><a><t><y>$ := [key=<_>ně,x=kr4,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=ins]]]];
<_><a><t><y>$ := [key=<_>a,x=kr5,cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=ins]]]];


{ pattern = stavení (nouns,neuter,-í) }
<_><í>$ := [key=<_>í,x=(st|ns),cat=[pos=n],
   morf=[infl=[pf=([gender=n,num=sg,case=(nom|gen|dat|acc|voc|loc)]
   | [gender=n,num=pl,case=(nom|gen|acc|voc)])]]];
<_><í><m>$ := [key=<_>í,x=(st|ns),cat=[pos=n],
   morf=[infl=[pf=([gender=n,num=sg,case=ins]
   | [gender=n,num=pl,case=dat])]]];
<_><í><c><h>$ := [key=<_>í,x=(st|ns),cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=loc]]]];
<_><í><m><i>$ := [key=<_>í,x=(st|ns),cat=[pos=n],
   morf=[infl=[pf=[gender=n,num=pl,case=ins]]]];


{ pattern = hrad (masc.  inanim.,‘‘hard’’ consonant) }
{ <T> is t or n,<K> is k,t or n }
<_>$ := [key=<_>,
```

```
   x=(hd1|hd1xx|hd2|hd2x|hd4|hd5x|hdus|hd1ek|hd1et|hd1en),
   cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=(nom|acc)]]]];
<_><u>$ := [key=<_>,x=(hd1|hd2|hd4|hd2x|hd5x|hd1xx),
   cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=(gen|dat|loc)]]]];
<_><u>$ := [key=<_>,x=(hd1g|hd1h|hd1ch|hd1k),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=(gen|dat|voc|loc)]]]];
<_><u>$ := [key=<_>us,x=hdus,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=(gen|dat|loc)]]]];
<_><T><u>$ :=
   [key=<_>e<T>,x=(hd1et|hd1en),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=(gen|dat|loc)]]]];
<_><k><u>$ := [key=<_>e<k>,x=hd1ek,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=(gen|dat|voc|loc)]]]];
<_><a>$ := [key=<_>,x=(hd4|hd5x),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=gen]]]];
<_><e>$ := [key=<_>,x=hd2,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=(voc|loc)]]]];
<_><e>$ :=
   [key=<_>,x=(hd1|hd2x|hd4|hd5x|hdus),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=voc]]]];
<_><e>$ := [key=<_>us,x=hdus,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=voc]]]];
<_><T><e>$ :=
   [key=<_>e<T>,x=(hd1en|hd1et),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=voc]]]];
<_><ě>$ := [key=<_>,x=(hd2x|hd5x),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=loc]]]];
<_><e><m>$ := [key=<_>,
   x=(hd1|hd2|hd4|hd2x|hd5x|hd1xx|hd1h|hd1g|hd1ch|hd1k),
   cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=ins]]]];
<_><e><m>$ := [key=<_>us,x=hdus,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=sg,case=ins]]]];
<_><K><e><m>$ :=
   [key=<_>e<K>,x=(hd1ek|hd1et|hd1en),cat=[pos=n],
```

```
   morf=[infl=[pf=[gender=i,num=sg,case=ins]]]];
<_><y>$ :=
   [key=<_>,x=(hd1|hd2|hd4|hd2x|hd5x|hd1xx|hd1h|hd1g|hd1ch|hd1k),
   cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=(nom|acc|voc|ins)]]]];
<_><y>$ := [key=<_>y,x=(hdpy|hdpky),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=(nom|acc|voc|ins)]]]];
<_><y>$ := [key=<_>us,x=hdus,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=(nom|acc|voc|ins)]]]];
<_><K><y>$ :=
   [key=<_>e<K>,x=(hd1ek|hd1et|hd1en),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=(nom|acc|voc|ins)]]]];
<_><ů>$ :=
   [key=<_>,x=(hd1|hd2|hd4|hd2x|hd5x|hd1xx|hd1h|hd1g|hd1ch|hd1k),
   cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=gen]]]];
<_><ů>$ := [key=<_>y,x=(hdpy|hdpky),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=gen]]]];
<_><ů>$ := [key=<_>us,x=hdus,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=gen]]]];
<_><K><ů>$ :=
   [key=<_>e<K>,x=(hd1ek|hd1et|hd1en),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=gen]]]];
<_><ů><m>$ :=
   [key=<_>,x=(hd1|hd2|hd4|hd2x|hd5x|hd1xx|hd1h|hd1g|hd1ch|hd1k),
   cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=dat]]]];
<_><ů><m>$ := [key=<_>y,x=(hdpy|hdpky),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=dat]]]];
<_><ů><m>$ := [key=<_>us,x=hdus,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=dat]]]];
<_><K><ů><m>$ :=
   [key=<_>e<K>,x=(hd1ek|hd1et|hd1en),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=dat]]]];
<_><e><c><h>$ :=
   [key=<_>,x=(hd1|hd2|hd4|hd2x|hd5x|hd1xx),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
```

```
<_><e><c><h>$ := [key=<_>y,x=hdpy,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><e><c><h>$ := [key=<_>us,x=hdus,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><T><e><c><h>$ :=
   [key=<_>e<T>,x=(hd1et|hd1en),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><k><á><c><h>$ := [key=<_>k,x=hd1k,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><k><á><c><h>$ := [key=<_>ek,x=hd1ek,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><k><á><c><h>$ := [key=<_>ky,x=hdpky,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><c><í><c><h>$ := [key=<_>k,x=hd1k,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><c><í><c><h>$ := [key=<_>ek,x=hd1ek,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><c><í><c><h>$ := [key=<_>ky,x=hdpky,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><í><c><h>$ := [key=<_>,x=hd1xx,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><š><í><c><h>$ := [key=<_>ch,x=hd1ch,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><z><í><c><h>$ := [key=<_>h,x=hd1h,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><z><í><c><h>$ := [key=<_>g,x=hd1g,cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
<_><á><c><h>$ :=
   [key=<_>,x=(hd1g|hd1h|hd1ch),cat=[pos=n],
   morf=[infl=[pf=[gender=i,num=pl,case=loc]]]];
```