Functional Morphology
User manual

Markus Forsberg
Department of Computing Science
Chalmers University of Technology
markus@cs.chalmers.se

June 27, 2004

1 Installation

- Download source files (`FM_LANG_v1.0.tgz`).
- Download lexicon file (`lang.lexicon`).
- Unpack the source: `gunzip FM_LANG_v1.0.tgz ; tar -xvf FM_LANG_v1.0.tar`.
- change directory: `cd ./FM_LANG_v1.0/`
- To compile FM (requires the Haskell GHC compiler): type: `make`
  This produces a binary `morpho_LANG`. If you have the program `strip` installed, type `strip morpho_LANG` to produce a smaller binary.
- To compile the c-trie (requires a C compiler):
  `make trie`
  This produces a binary `tools/trie`.
- To compile the lexicon cleaner (requires the Haskell GHC compiler):
  `make cleanLex`
  This produces a binary `tools/cleanLex`.

2 Running the program

We assume that you have downloaded the tutorial language, the Latin morphology, and compiled in the manner described above. The other languages are handled in an analogous way.

To run the program, the external lexicon, `latin.lexicon`, needs to either be in the same directory as where the program is run or pointed to with the environment variable `FM_LAT`. E.g., in Bash, we could give the following command to point the lexicon *latin.lexicon* in the directory `/home/markus/FM`:

```
$ export FM_LAT="/home/markus/FM/latin.lexicon"
```

The program parameters are listed by giving the help command `morpho_lat -h`.

```
$ ./morpho_lat -h

 |--------------------------------------|
 |        Program parameters            |
 |--------------------------------------|
 | -h              | Display this message |
 |--------------------------------------|
 | <None>          | Enter tagger mode    |
 |--------------------------------------|
 | -s              | Enter interactive    |
 |                 | synthesiser mode     |
 |--------------------------------------|
 | -i              | Enter inflection     |
 |                 | mode                 |
 |--------------------------------------|
 | -lex    [file] | Full form lexicon    |
 | -tables [file] | Tables               |
 | -gf     [file] | GF source code       |
 | -latex  [file] | LaTeX source code    |
 | -xml    [file] | XML source code      |
 | -lexc   [file] | LexC source code     |
 | -xfst   [file] | XFST source code     |
 | -sql    [file] | SQL source code      |
 |--------------------------------------|
```

The meaning of the different parameters will be explained below.

2.1 Tagger mode

Tagging is the process of analysing words into their dictionary form and their grammatical description.

Let us tag some words, **servi** and **servusne**.

```
 $ ./morpho_lat

********************************************
* Latin Morphology                         *
********************************************
* Functional Morphology v1.00              *
* (c) Markus Forsberg & Aarne Ranta 2004   *
* under GNU General Public License.        *
********************************************


Dictionary loaded: DF = 196 and WF = 8131.

servi

[ <servi>
1. servus (103) Noun - Plural Vocative - Masculine
2. servus (103) Noun - Plural Nominative - Masculine
3. servus (103) Noun - Singular Genitive - Masculine
]

servusne

[ <servusne>
1. Composite: servus (103) Noun - Singular Nominative - Masculine |
              ne (22) Particle - Invariant -
]
```

The `DW count` is the number of dictionary words in the lexicon, and `WF count` is the number of word forms.

2.2 Synthesiser mode

The synthesiser generates from a word form the inflection table which the word is a member of. If the word form is a member of many inflection tables, all tables are listed. The synthesiser handles no composite analysis.

```
$ ./morpho_lat -s
********************************************
* Latin Morphology                         *
********************************************
* Functional Morphology v1.00              *
* (c) Markus Forsberg & Aarne Ranta 2004   *
* under GNU General Public License.        *
********************************************

[Synthesiser mode]

Enter a Latin word in any form.

If the word is not found, a [command] with [arguments].

Type 'c' to list commands.

Type 'q' to quit.

Dictionary loaded: DF = 196 and WF = 8131.

> puellae
puella
Noun
Feminine
Singular Nominative: puella
Singular Vocative: puella
Singular Accusative: puellam
....
```

2.3 Inflection mode

In inflection mode, you can generate inflection tables that is not part of the lexicon, by giving a interface function and a dictionary form. The inflection mode is the same as synthesis mode with no lexicon loaded. The inflection mode is for lexicographers who intend to extend the lexicon.

```
$ ./morpho_lat -i
********************************************
* Latin Morphology                         *
********************************************
* Functional Morphology v1.00              *
* (c) Markus Forsberg & Aarne Ranta 2004   *
* under GNU General Public License.        *
********************************************



[Inflection mode]

Enter [command] [dictionary form].

Type 'c' to list commands.

Type 'q' to quit.

> c
d1puella rosa
d1puellaMasc poeta
d2bellum bellum
```

```
d2liber liber
d2puer puer
d2servus servus
...
```

2.4 Translators

Functional morphology supports generation of a number of formats:

- Full form lexicon
- Inflection tables, ASCII and Latex
- Grammatical Framework source code
- XML representation of the lexicon
- XFST source code
- LexC source code
- SQL source code

2.4.1 Full form lexicon

The full form lexicon format is a listing of all word forms in the lexicon, with its analysis.

```
$ ./morpho_lat -lex
a:a (107) Preposition - Invariant - (0)
ab:ab (108) Preposition - Invariant - (0)
absque:absque (109) Preposition - Invariant - (0)
accola:accola (51) Noun - Singular Ablative - Masculine (0)
accola:accola (51) Noun - Singular Vocative - Masculine (0)
...
```

2.4.2 Inflection tables

The are two ways of generating the inflection tables of the lexicon, as text or as LaTeX source code.

```
$ ./morpho_lat -tables
gladius
Noun
Masculine
Singular Nominative: gladius
Singular Vocative: gladi
Singular Accusative: gladium
Singular Genitive: gladi
Singular Dative: gladio
Singular Ablative: gladio
Plural Nominative: gladii
Plural Vocative: gladii
Plural Accusative: gladios
Plural Genitive: gladiorum
Plural Dative: gladiis
Plural Ablative: gladiis
...
```

Type `./morpho_sw -latex` to generate LaTeX.

2.4.3 Grammatical framework (GF)

The generated GF source format:

```
$ ./morpho_lat -gf
...
cat Noun;

fun gladius_0 : Noun ;

lin gladius_0 = {s = table {
  Singular Nominative => "gladius" ;
  Singular Vocative => "gladi" ;
  Singular Accusative => "gladium" ;
  Singular Genitive => "gladi" ;
  Singular Dative => "gladio" ;
  Singular Ablative => "gladio" ;
  Plural Nominative => "gladii" ;
  Plural Vocative => "gladii" ;
  Plural Accusative => "gladios" ;
  Plural Genitive => "gladiorum" ;
  Plural Dative => "gladiis" ;
  Plural Ablative => "gladiis" };
  h1 = Masculine
} ;
...
```

2.4.4 XML format

A XML representation of the dictionary is produced by the following command:

```
$ ./morpho_lat -xml

<?xml version="1.0"?>
<lexicon>
 <class category="Verb">
  <lexicon_entry>
   <dictionary_form value="adflare" />
   <inflection_table>
    <inflection_form pos="Indicative First Singular Present Active">
     <variant word="adflo" />
    </inflection_form>
    <inflection_form pos="Indicative Second Singular Present Active">
...
```

2.4.5 XFST format

The Xerox Finite State Transducer format is produced by the following command:

```
$ ./morpho_lat -xfst
define Noun [
  [ {gladius} %+Singular  %+Nominative  %+Masculine .x. {gladius}]  |
  [ {gladius} %+Singular  %+Vocative  %+Masculine .x. {gladi}]  |
  [ {gladius} %+Singular  %+Accusative  %+Masculine .x. {gladium}]  |
  [ {gladius} %+Singular  %+Genitive  %+Masculine .x. {gladi}]  |
  [ {gladius} %+Singular  %+Dative  %+Masculine .x. {gladio}]  |
  [ {gladius} %+Singular  %+Ablative  %+Masculine .x. {gladio}]  |
  [ {gladius} %+Plural  %+Nominative  %+Masculine .x. {gladii}]  |
  [ {gladius} %+Plural  %+Vocative  %+Masculine .x. {gladii}]  |
  [ {gladius} %+Plural  %+Accusative  %+Masculine .x. {gladios}]  |
  [ {gladius} %+Plural  %+Genitive  %+Masculine .x. {gladiorum}]  |
  [ {gladius} %+Plural  %+Dative  %+Masculine .x. {gladiis}]  |
  [ {gladius} %+Plural  %+Ablative  %+Masculine .x. {gladiis}]
  |
  [ {filius} %+Singular  %+Nominative  %+Masculine .x. {filius}]  |
  [ {filius} %+Singular  %+Vocative  %+Masculine .x. {fili}]  |
...
```

2.4.6 LexC format

The Xerox LexC format.

```
$ ./morpho_lat -lexc
LEXICON Root

! category Verb

adflo:adflare+Indicative+First+Singular+Present+Active # ;
adflas:adflare+Indicative+Second+Singular+Present+Active # ;
adflat:adflare+Indicative+Third+Singular+Present+Active # ;
adflamus:adflare+Indicative+First+Plural+Present+Active # ;
adflatis:adflare+Indicative+Second+Plural+Present+Active # ;
adflant:adflare+Indicative+Third+Plural+Present+Active # ;
adflabam:adflare+Indicative+First+Singular+Imperfect+Active # ;
adflabas:adflare+Indicative+Second+Singular+Imperfect+Active # ;
...
```

2.4.7 SQL format

A SQL database containing the database can be generated as follows:

```
$ ./morpho_lat -sql

CREATE TABLE LEXICON
(
ID INTEGER NOT NULL,
DICTIONARY VARCHAR(50) NOT NULL,
CLASS VARCHAR(50) NOT NULL,
WORD VARCHAR(50) NOT NULL,
POS VARCHAR(50) NOT NULL);

INSERT INTO LEXICON VALUES ('1','gladius','Noun','gladius','Singular Nominative - Masculine');
```

2.5 Extending the lexicon

The lexicon can easily be extended by adding entries to the external lexicon `latin.lexicon`.

However, the interface functions may be incomplete, so to handle uncommon, irregular cases, you may need to define a new paradigm in the Haskell modules. See the tutorial for more information on extending the set of paradigms.

3 Other tools

3.1 C-Trie

If you really need speed in your analysis, or for any other reason, you may use the trie implemented in C. However, in this version, no composite analysis is performed.

To use the C-trie, first print out a fullform lexicon from a morphology, e.g. Latin: `morpho_lat -lex latin.fullform`, and give it as argument to the C-trie program: `tools/trie latin.fullform`.

```
$ trie latin.fullform

-----------------------------------------------------
|              C-Trie 0.1, April 2004               |
|---------------------------------------------------|
|             Author: Markus Forsberg               |
|                                                   |
|   Send bug reports to: markus@cs.chalmers.se      |
-----------------------------------------------------

[Reading lexicon read from 'latin.fullform'...]

........

[ Number of entries :      8k entries ]
[ Build CPU time    :    0.14 seconds ]

puella
puella:
        puella (47) Noun - Singular Nominative - Feminine (0)
        puella (47) Noun - Singular Vocative - Feminine (0)
        puella (47) Noun - Singular Ablative - Feminine (0)
```

3.2 Lexicon cleaner

The lexicon cleaner is a small program that sorts and removes duplicates of a lexicon.

Usage example: `cleanLex latin.lexicon`
This produces a new file `latin.lexicon.new`, which is the input lexicon sorted and with no duplicates.