Final Report

Implementation Details

Lexicon File (lexc)

For the implementation of German adjective endings, the following multicharacter symbols were used in order to correctly mark each adjective for number, case, gender, and type of declension being used:

1. **Number**
    1.1. Singular: `+Sg`
    1.2. Plural: `+Pl`
2. **Case**
    2.1. Nominative: `+Nom`
    2.2. Accusative: `+Acc`
    2.3. Dative: `+Dat`
    2.4. Genitive: `+Gen`
3. **Gender**
    3.1. Masculine: `+Masc`
    3.2. Feminine: `+Fem`
    3.3. Neuter: `+Neu`
4. **Type of declension**
    4.1. Strong declension: `+S`
    4.2. Weak declension: `+W`
    4.3. Mixed declension: `+M`
    4.4. Bare stem (no adjective declension): `+Bare`
5. **Other symbols used for the implementation**
    5.1. Adjective: `+Adj`

The lexicon root was defined as *Adjectives*. Only a small number of adjectives were included to demonstrate the functionality of the finite state transducer, however care was taken to include all possible stem changes that can occur to German adjectives when undergoing declension. A total of eleven adjectives were chosen for the purpose of this demonstration, with five of the total eleven requiring a replace rule to implement a change on the adjective stem. The adjectives chosen, along with their English translations, are as follows:

1. blau – blue
2. ***dunkel*** – dark
3. ***edel*** – noble
4. gut – good
5. groß – big, great
6. grün – green
7. ***hoch*** – high
8. klein – small
9. schön – beautiful
10. ***teuer*** – expensive
11. ***trocken*** – dry

Adjectives in **bold** undergo a stem change when declined – see regex for more details.

The *+Adj* (adjective) multicharacter symbol is immediately affixed to each of the above listed adjectives' lexical (upper) sides. This takes place in the lexicon *Adjectives*. In addition, the surface (lower) side of each of the words is set as the stem, so the form of *gut,* for example, would be: gut+Adj:gut

Every element in the lexicon is also assigned the continuation class *Ending; This* continuation class contains all of the possible combinations for number, gender, case, and type of adjective declension. The morphological form of the adjective ending is determined by these four aspects. Those four aspects are appended to the lexical side of each element. The appropriate adjective ending is also appended to the surface side of each element.

The implementation produces a total of forty-nine different adjective forms for each element of the lexicon (please note many of these endings have an identical morphological form, however they differ by the particular combination of number, case, gender, and type of declension – this information is shown on the lexical (upper) side). For this reason, comments have been utilised to increase ease of reading of the lexicon file. The file has been laid out with a hierarchical structure. The type of adjective declension, followed by the three possible genders and plural forms. The bare stem form also appears at the end.

See code appendix for the complete lexc file.

Regular Expression File (regex)

The regular expression file, called *rules.regex*, contains a total of four replace rules. A number of German adjectives "ending in -*el*, -*er* (and, optionally, those in -*en*) most often drop the -e- when an ending is added" (Durrell, 2015). Furthermore, "the adjective *hoch* 'high' has the special form *hoh*- when endings are added". These four exceptions are coded into regular expressions which follow:

1. `[ e l -> l || _ \.#. ]`
2. `[ e r -> r || _ \.#. ]`
3. `[ e n -> n || _ \.#. ]`
4. `[ h o c h -> h o h || _ \.#. ]`

The regular expressions 1-3 function in a similar way. For an explanation of 1-3, the regular expression 1 will be used for reference. The rule searches through the lexicon and applies the rule to all elements which contain the substring *'el'*, where the substring does not occur at the end of word, the substring *'el'* will be replaced with just *'l'*, therefore dropping the -e- from the adjective. The backslash character is used to signify NOT, this character in combination with the word final symbol .#. means not word final. Implementing the rule this way means that the word ending in *'el'* will be printed in its full proper form when in its bare form, but the -e- will be dropped whenever an ending is appended.
The above explanation also applies to the replace rules 2 and 3, above.

The regular expression 4 works in a similar way, except the entire stem of the adjective *hoch* is replaced with the special form *hoh*- when an adjective ending is present.
See code appendix for the complete regex file.

Problems Encountered

A number of problems were encountered when attempting to implement the project. In order to overcome these problems, a few things had to be changed from the original proposal.
The original idea was to also include a number of determiners – definite and indefinite articles – and to check the preceding determiner (or the lack of a determiner) in order to decide what type of declension to use (weak, strong, or mixed). This implementation proved

too difficult to carry out given the scope of this project. To overcome this problem a multicharacter symbol was assigned to each form of declension. This way, every type of declension is still encoded in the lexicon file and if the project were to be expanded on, a method to check the determiner could be designed which would add the multicharacter symbol after the type of declension is identified.

There was also a couple of problems encountered when designing regular expressions which produced the desired outcome. However, after some research in *Finite-State Morphology: Xerox Tools and Techniques (2003)* and trial and error, four regular expressions were designed which output all the adjective endings correctly. This means the adjective is printed correctly in its bare stem form and a stem change occurs only when an ending is present (this has already been explained in detail above).

Evaluation of Coverage

An German language article taken from *der Spiegel* (1976) was used to test the coverage of the finite state transducer. The article was pasted into a text file and every space was replaced with a new line character. Any punctuation was also removed. The total size of the word list was 8145 words. The finite state transducer was run on the word list and was successful 49 times (0.60%). The transducer functioned as desired and found adjectives present in the word list regardless of the ending attached to the adjective. The transducer returned a list of the possible lexical forms of the word when it was found. For example, when *große* was found, the following was returned:

```
große       groß+Adj+Pl+Acc+S
große       groß+Adj+Pl+Nom+S
große       groß+Adj+Sg+Masc+Nom+W
große       groß+Adj+Sg+Fem+Acc+W
große       groß+Adj+Sg+Fem+Acc+S
große       groß+Adj+Sg+Fem+Acc+M
große       groß+Adj+Sg+Fem+Nom+W
große       groß+Adj+Sg+Fem+Nom+S
große       groß+Adj+Sg+Fem+Nom+M
große       groß+Adj+Sg+Neu+Acc+W
große       groß+Adj+Sg+Neu+Nom+W
```

The surface side of the adjective is shown on the left hand side, and the lexical side is shown on the right. This evaluation shows proof of concept; if a large corpus adjectives was added to the lexicon file, the percentage of found words would increase significantly.

(***Command used:*** `$ cat wordlist.txt | ./lookup adjs_german.fst`)

Conclusion

Overall, the final project achieves the main goals set out in the original proposal. The lexicon and regular expressions which have been implemented work together using Xerox tools to display all possible German adjective endings in both their lexical and surface forms. Although the original proposal had to be tweaked somewhat to reflect the scope of the project, the central task has still been implemented in full. The project displays a competent command of the finite state lexicon compiler being used, *lexc*, and of regular expressions, which can be seen in the replace rules. A simple script is also included in the finished project. This script can be run in the terminal and will automatically concatenate the lexicon with the rules. The script then composes the network and prints both the upper and lower sides of every element in the lexicon.

Finally, if the project were to be designed from the beginning again, three of replace rules seen in the regular expressions file could be condensed into a single rule. More research would need to be carried out into regular expressions but given the similarity of the rules it is very likely a simpler rule could be designed which would encompass all three. An OR rule could be used in the regular expression but memory would be needed to store which option from the OR rule should be used to then assign the correct corresponding replacement.

Apart from this possible improvement, the rest of the project was implemented in an efficient way which could easily be expanded to include more aspects of the German language in the finite state transducer.

Code Appendix

Lexicon File (lexc)

Filename: *adjectives.lexc*

```
Multichar_Symbols
+Adj +Sg +Pl +Bare    ! Adjective, Singular, Plural, Bare Stem
+Masc +Fem +Neu       ! Masculine, Feminine, Neuter (Genders)
+Nom +Acc +Dat +Gen   ! Nominative, Accusative, Dative, Genetive (Cases)
+W +S +M              ! Weak, Strong, Mixed (Adjective Declension)

Lexicon Root
  Adjectives;

Lexicon Adjectives
  gut+Adj:gut            Ending;
  klein+Adj:klein        Ending;
  schön+Adj:schön        Ending;
  blau+Adj:blau          Ending;
  grün+Adj:grün          Ending;
  groß+Adj:groß          Ending;
  trocken+Adj:trocken    Ending; ! irregular - stem change
  edel+Adj:edel          Ending; ! irregular - stem change
  dunkel+Adj:dunkel      Ending; ! irregular - stem change
  hoch+Adj:hoch          Ending; ! irregular - stem change
  teuer+Adj:teuer        Ending; ! irregular - stem change
                         ! (see rules.regex for replace rules)

Lexicon Ending
! WEAK MASCULINE
+Sg+Masc+Nom+W:e         #;
+Sg+Masc+Acc+W:en        #;
+Sg+Masc+Dat+W:en        #;
+Sg+Masc+Gen+W:en        #;

! WEAK FEMININE
+Sg+Fem+Nom+W:e          #;
+Sg+Fem+Acc+W:e          #;
+Sg+Fem+Dat+W:en         #;
+Sg+Fem+Gen+W:en         #;

! WEAK NEUTER
+Sg+Neu+Nom+W:e          #;
+Sg+Neu+Acc+W:e          #;
+Sg+Neu+Dat+W:en         #;
+Sg+Neu+Gen+W:en         #;

! WEAK PLURAL
+Pl+Nom+W:en             #;
+Pl+Acc+W:en             #;
+Pl+Dat+W:en             #;
+Pl+Gen+W:en             #;

!-----------------------

! STRONG MASCULINE
+Sg+Masc+Nom+S:er        #;
+Sg+Masc+Acc+S:en        #;
+Sg+Masc+Dat+S:em        #;
+Sg+Masc+Gen+S:en        #;
```

```
! STRONG FEMININE
+Sg+Fem+Nom+S:e          #;
+Sg+Fem+Acc+S:e          #;
+Sg+Fem+Dat+S:en         #;
+Sg+Fem+Gen+S:en         #;

! STRONG NEUTER
+Sg+Neu+Nom+S:es         #;
+Sg+Neu+Acc+S:es         #;
+Sg+Neu+Dat+S:em         #;
+Sg+Neu+Gen+S:en         #;

! STRONG PLURAL
+Pl+Nom+S:e              #;
+Pl+Acc+S:e              #;
+Pl+Dat+S:en             #;
+Pl+Gen+S:er             #;

!------------------------

! MIXED MASCULINE
+Sg+Masc+Nom+M:er        #;
+Sg+Masc+Acc+M:en        #;
+Sg+Masc+Dat+M:en        #;
+Sg+Masc+Gen+M:en        #;

! MIXED FEMININE
+Sg+Fem+Nom+M:e          #;
+Sg+Fem+Acc+M:e          #;
+Sg+Fem+Dat+M:en         #;
+Sg+Fem+Gen+M:en         #;

! MIXED NEUTER
+Sg+Neu+Nom+M:es         #;
+Sg+Neu+Acc+M:es         #;
+Sg+Neu+Dat+M:en         #;
+Sg+Neu+Gen+M:en         #;

! MIXED PLURAL
+Pl+Nom+M:en             #;
+Pl+Acc+M:en             #;
+Pl+Dat+M:en             #;
+Pl+Gen+M:en             #;

!------------------------

! BARE STEM
+Bare:0                  #;

END
```

Regular Expressions File (regex)

Filename: *rules.regex*

```
# Replace Rules
[ e l -> l || _ \.#. ] # el changes to l when an ending is present
.o.
[ e r -> r || _ \.#. ] # er changes to r when an ending is present
.o.
[ e n -> n || _ \.#. ] # en changes to n when an ending is present
.o.

#special case for 'hoch': becomes 'höh-' when an ending is present
[ h o c h -> h o h || _ \.#. ];
```

Script File

Filename: *adjs_german.script*

```
echo << BUILDING FST >>
clear stack
read lexc < adjectives.lexc
define LEX
read regex < rules.regex
define RUL
read regex LEX .o. RUL;
echo << LEXICAL (UPPER) SIDE >>
print upper
echo << SURFACE (LOWER) SIDE >>
print lower
save adjs_german.fst
echo << END OF SCRIPT >>
```

Bibliography

Beesley, K. R., & Karttunen, L. (2003). Finite-state morphology: Xerox tools and techniques (pp. 81-298). CSLI, Stanford.

Der Spiegel (1976, 02 Feb.). Article retrieved from: http://www.spiegel.de/spiegel/print/d-41279330.html