Finite-state morphology

We have seen how to handle morphology with FSTs.
We will step back & formally characterize morphological operations, focusing on composition:

- Composition handles concatenative morphology cleanly
- Composition handles:
  - restrictions on the kinds of bases that affixes can attach to
  - modifications on the bases that affixes attach to

Material is adapted from Roark & Sproat (2007), Computational Approaches to Morphology and Syntax, esp. ch. 2

Example of Latin

Latin scripserunt is a combination of:

- stem scrip- (‘write’), which becomes scrip- before /s/
- perfect stem-forming -s- (for third conjugation verbs)
- (perfect) third person plural suffix erunt

Morphological analysis: i) detect structure of word forms, and ii) relate word forms
- detect structure:
  scrip+sperfect+eruntthird_plural_active_indicative
  - We will use the function D to represent this step
- relate to canonical form (lemmatization):
  scriboperfect_third_plural_active_indicative
  - We can use a function L to obtain lemma from decomposed form (structure)
  - i.e., \( D \circ L \)

Syntagmatic variation

Simple concatenation

Given a stem A and a suffix \( \beta \), we can create a form \( \Gamma \) with regular concatenation:

1. \( \Gamma = A \cdot \beta \)

What if instead we have a function \( \beta' \) which takes a string as input & outputs a string concatenated with \( \beta \)?

2. \( \beta' = \Sigma^* [\epsilon \cdot \beta] \)
   - \( \Sigma = \text{alphabet of symbols} \)
   - \( \Sigma^* \) is used here to specify a regular relation which maps strings into themselves

Now, we have:

3. \( \Gamma = A \circ \beta' \)

Syntagmatic variation

Prosodically Governed Concatenation

Some affixes have prosodic conditions, e.g., comparative -er and superlative -est in English:

- Generally speaking: any attach to monosyllabic or disyllabic stems
- The base/stem can be characterized as:

4. \( B = C \cdot VC' (VC') \)
- and the affix as:

5. \( \kappa = B [\epsilon : er[+\textit{COMP}]] \)
- resulting in:

6. \( \Gamma = A \circ \kappa \)
   - The only non-null \( \Gamma \) cases will be the ones where the base of \( A \) matches \( B \)
This is easily captured by defining a function \( \text{bad} \) Insert a marker (\( '>' \)) for where the infix goes 1. Insert a marker (\( '>' \)) for where the infix goes 2. Convert the marker to the affix (e.g., -um-)

Consider infixes like -um- in Filipino languages, e.g., Bontoc
- Ignores the onset sound of the word and prefixes to the remainder of the word
  - \( \text{antj} \)‘tall’: \( \text{umanjt} \)‘I am getting taller’
  - \( \text{k} \)‘good’: \( \text{kum} \)‘I am getting better’
- Multiple infixes attach in this same spot, so it makes sense to break this down into 2 parts:
  1. Insert a marker (\( '>' \)) for where the infix goes
  2. Convert the marker to the affix (e.g., -um-)

Syntagmatic variation
Extrametrical infixation

Consider infixes like -um- in Filipino languages, e.g., Bontoc
- Ignores the onset sound of the word and prefixes to the remainder of the word
  - \( \text{antj} \)‘tall’: \( \text{umanjt} \)‘I am getting taller’
  - \( \text{k} \)‘good’: \( \text{kum} \)‘I am getting better’
- Multiple infixes attach in this same spot, so it makes sense to break this down into 2 parts:
  1. Insert a marker (\( '>' \)) for where the infix goes
  2. Convert the marker to the affix (e.g., -um-)

Syntagmatic variation
Subsegmental morphology

Two-step insertion process:
1. Marker transducer \( M \): insert \( '>' \) at appropriate spot
   \( M = C?\epsilon : > \Sigma^* \)
2. Infixation transducer \( i \): map \( '>' \) to -um-

Precompose these two steps:
\( \mu = M \circ i \)

Meaning that a final word form is:
\( \Gamma = A \circ \mu \)

Syntagmatic variation
Root-and-pattern morphology

Arabic verbs (derivational morphology):
- consonantal roots
- prosodic shape given by a prosodic template
- particular vowels chosen by intended aspect
  (perfect/imperfect)

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Template</th>
<th>Verb stem</th>
<th>Gloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>C_{1}aC_{2}aC_{3}</td>
<td>katab</td>
<td>'wrote'</td>
</tr>
<tr>
<td>II</td>
<td>C_{1}aC_{2}aC_{3}</td>
<td>katab</td>
<td>'caused to write'</td>
</tr>
<tr>
<td>III</td>
<td>C_{1}aC_{2}aC_{3}</td>
<td>katab</td>
<td>'corresponded'</td>
</tr>
<tr>
<td>VII</td>
<td>nCr_{1}aC_{2}aC_{3}</td>
<td>nkatab</td>
<td>'subscribed'</td>
</tr>
</tbody>
</table>

To obtain a transducer for all these templates:
\( \tau = \bigcup_{p \in \text{patterns}} \tau_{p} \)
Syntagmatic variation

Root-and-pattern morphology (3)

Need a transducer to link the root to the templates:

- Allow for optional vowels between consonants:

\[ \lambda_1 = C[\epsilon : V]^\prime \cdot C[\epsilon : V]^\prime \cdot C \]

- Allow for doubling of center consonant (pattern II) . . . need general rewrite rules:

\[ \lambda_2 : C_i \rightarrow C_iC_i \]

(26) \( \lambda = \lambda_1 \circ \lambda_2 \)

Derive forms:

(28) \( \Gamma = P \circ \lambda \circ \tau \)

One can also compile \( \lambda \circ \tau \) into its own “pattern” machine

Paradigmatic variation

A paradigm is an array where each cell corresponds to a bundle of features

- It characterizes how morphologically complex forms relate to one another
- e.g., Latin nouns, declension 1 (F)

<table>
<thead>
<tr>
<th></th>
<th>Singular</th>
<th>Plural</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominative</td>
<td>femina</td>
<td>feminarum</td>
</tr>
<tr>
<td>Genitive</td>
<td>feminae</td>
<td>feminis</td>
</tr>
<tr>
<td>Dative</td>
<td>feminam</td>
<td>feminas</td>
</tr>
<tr>
<td>Accusative</td>
<td>femina</td>
<td>feminis</td>
</tr>
<tr>
<td>Ablative</td>
<td>femina</td>
<td>feminis</td>
</tr>
</tbody>
</table>

There are regularities which seem to argue for a first-class status of paradigms
- e.g., ablative & dative plurals

Paradigmatic variation

A Computational Characterization

1. Relate morphosyntactic features to abstract morphomic features (transducer \( \alpha \))

- NEUT NOM U ACC SG \( \rightarrow \) NEUTNASG
- NEUT NOM U ACC PL \( \rightarrow \) NEUTNAPL
- NOM SG \( \rightarrow \) NomSG
- GENDER DAT PL \( \rightarrow \) DATABLPL
- GENDERABL PL \( \rightarrow \) DATABLPL

... (25) \( \alpha_1 = C[\epsilon : V]^\prime \cdot C[\epsilon : V]^\prime \cdot C \)

2. Relate morphomic forms to particular surface forms (for a particular word class) (transducer \( \sigma \))

- \( \Sigma^* [\text{I-II DATABLPL : is}] \)
- \( \Sigma^* [\text{NEUTNAPL : a}] \)
- \( \Sigma^* [\text{I-II NEUTNASG : um}] \)
- \( \Sigma^* [\text{III DATABLPL : ibus}] \)

... (26) \( \lambda_2 : C_i \rightarrow C_iC_i \)

... (27) \( \lambda = \lambda_1 \circ \lambda_2 \)

... (28) \( \Gamma = P \circ \lambda \circ \tau \)

Could also precompile \( \sigma' = \alpha \circ \sigma \), thereby hiding the abstraction

Paradigmatic variation

Reduplication

Reduplication involves potentially unbounded copying
- Copying not allowed by strict FSTs
- Bounded copying—however inelegantly—can be handled by FSTs

Gothic past tense of Class VII verbs

<table>
<thead>
<tr>
<th>Infinitive</th>
<th>Gloss</th>
<th>Preterite</th>
</tr>
</thead>
<tbody>
<tr>
<td>hal'dan</td>
<td>hold</td>
<td>hálhálad</td>
</tr>
<tr>
<td>ga-stal'dan</td>
<td>possess’</td>
<td>ga-staistal'd</td>
</tr>
<tr>
<td>af-ál'kan</td>
<td>‘deny’</td>
<td>af-ái'ák</td>
</tr>
<tr>
<td>sle'pan</td>
<td>‘sleep’</td>
<td>saislep</td>
</tr>
</tbody>
</table>

Reduplication (2)

Rule:
- Prefix syllable \( (A)\text{Ca}^i \) to the stem
  - \( C \) is a consonant position
  - \( (A) \) is an optional appendix position
- Copy the onset of the stem to the \( C \) position
  - If there is a pre-onset appendix \( /s/ \) (i.e., \( /s/ \) before \( /p,t,k/ \)), copy to the \( (A) \) position

The transducer for this simply hard-codes the proper sequences to obtain copying
- e.g., 1) \( \epsilon:h \) arc, 2) \( \epsilon:a'i \) arc, 3) \( h:h \) arc
Unbounded Reduplication

Consider Bambara noun reduplication:

(30) \textit{wulu o wulu}  
\begin{itemize}
  \item \textit{dog MARKER dog}
  \item ‘whichever dog’
\end{itemize}

(31) \textit{wulu-nyinina o wulu-nyinina}  
\begin{itemize}
  \item \textit{dog searcher MARKER dog searcher}
  \item ‘whichever dog searcher’
\end{itemize}

(32) \textit{malo-nyinina-filéla o}  
\begin{itemize}
  \item \textit{rice searcher watcher MARKER}
  \item \textit{malo-nyinina-filéla}
  \item \textit{rice searcher watcher}
  \item ‘whichever rich searcher watcher’
\end{itemize}

▶ The morpheme \textit{o} in principle is unbounded
▶ Cannot simply hard-code material before/after \textit{o}

Unbounded Reduplication (2)

Think of reduplication as two components:

1. Prosodic constraints: e.g., make sure reduplicated material is of form \((A)Ca’i\)
   ▶ This can be handled with regular finite-state operations
2. Copying component: verify that the prefix matches the base

Unbounded Reduplication (3)

For Gothic, assume transducer \(R\), which composes with a base \(\beta\) and adds indices to elements in prefix and base

(33) \(\alpha = \beta \circ R = (A_1)C_2a’s\)

Input stem \textit{skäip} will result in the output \(X_1X_2aıs_1kւäip\)

▶ \(X\) ranges over possible segments
▶ An additional component checks whether \(X\) is well-formed, i.e., indices match