Semitic Morphological Processing Using Finite State Transducers with Feature Structure Weights

Michael Gasser
Overview
Overview

• Finite state morphology
Overview

- Finite state morphology
- Tigrinya verb morphology
Overview

- Finite state morphology
- Tigrinya verb morphology
- The challenge of non-concatenative morphology
Overview

- Finite state morphology
- Tigrinya verb morphology
- The challenge of non-concatenative morphology
- Weighted finite state transducers
Overview

• Finite state morphology
• Tigrinya verb morphology
• The challenge of non-concatenative morphology
• Weighted finite state transducers
• Feature structures and unification
Overview

- Finite state morphology
- Tigrinya verb morphology
- The challenge of non-concatenative morphology
- Weighted finite state transducers
- Feature structures and unification
- FSTs weighted with feature structures
Overview

- Finite state morphology
- Tigrinya verb morphology
- The challenge of non-concatenative morphology
- Weighted finite state transducers
- Feature structures and unification
- FSTs weighted with feature structures
- Weighted FSTs applied to Tigrinya verbs
Morphological processing
Morphological processing

- Surface and lexical levels
Morphological processing

• Surface and lexical levels
• boxes ⇔ box+s
Morphological processing

- **Surface and lexical levels**
- $boxes \Leftrightarrow box+s$
- $boxes \Leftrightarrow box+\text{PLUR}$
Morphological processing

- **Surface and lexical levels**
- $\text{boxes} \leftrightarrow \text{box+s}$
- $\text{boxes} \leftrightarrow \text{box+PLUR}$
- Morphotactics
Morphological processing

- Surface and lexical levels
- \( \text{boxes} \Leftrightarrow \text{box}+s \)
- \( \text{boxes} \Leftrightarrow \text{box}+\text{PLUR} \)
- Morphotactics
- Phonological, graphological rules (alternation rules)
Morphological processing

• Surface and lexical levels
• boxes ⇔ box+s
• boxes ⇔ box+PLUR
• Morphotactics
• Phonological, graphological rules (alternation rules)
  - The old way: context-sensitive rewrite rules
    + → e / S_s; S = {s, z, h, x}
Morphological processing

- **Surface and lexical levels**
- **boxes** $\Rightarrow$ **box+s**
- **boxes** $\Rightarrow$ **box+PLUR**
- **Morphotactics**
- **Phonological, graphological rules (alternation rules)**
  - The old way: context-sensitive rewrite rules
    - $+ \rightarrow e / S_s$; $S = \{s, z, h, x\}$
      - Context-sensitive
Morphological processing

• Surface and lexical levels
• $boxes \Leftrightarrow box+s$
• $boxes \Leftrightarrow box+\text{PLUR}$
• Morphotactics
• Phonological, graphological rules (alternation rules)
  - The old way: context-sensitive rewrite rules
    $+ \rightarrow e / S_s$; $S = \{s, z, h, x\}$
    - Context-sensitive
    - Ordered
Morphological processing

- **Surface and lexical levels**

- **boxes** $\Leftrightarrow$ **box+s**

- **boxes** $\Leftrightarrow$ **box+PLUR**

- **Morphotactics**

- **Phonological, graphological rules (alternation rules)**
  - The old way: context-sensitive rewrite rules
    + $+ \rightarrow e / S\_s; S = \{s, z, h, x\}$
      - Context-sensitive
      - Ordered
      - Uni-directional
Finite state morphology
Finite state morphology

- Each alternation rule takes the form of a finite state transducer
Finite state morphology
Finite state morphology

- FSTs are invertible: what works for analysis also works for generation
Finite state morphology

- FSTs are invertible: what works for analysis also works for generation
- Doing away with rule ordering
Finite state morphology

- FSTs are invertible: what works for analysis also works for generation
- Doing away with rule ordering
  - Two-level morphology (Koskenniemi)
Finite state morphology

• FSTs are invertible: what works for analysis also works for generation
• Doing away with rule ordering
  - Two-level morphology (Koskenniemi)
    - Surface and lexical levels related directly by an intersection of parallel FSTs, one for each rule
Finite state morphology

• FSTs are invertible: what works for analysis also works for generation
• Doing away with rule ordering
  - Two-level morphology (Koskenniemi)
    • Surface and lexical levels related directly by an intersection of parallel FSTs, one for each rule
  - Composition of rule FSTs (Karttunen et al.)
Finite state morphology

• FSTs are invertible: what works for analysis also works for generation

• Doing away with rule ordering
  - Two-level morphology (Koskenniemi)
    • Surface and lexical levels related directly by an intersection of parallel FSTs, one for each rule
  - Composition of rule FSTs (Karttunen et al.)
    • $f_1: s_1 \Rightarrow s_2$
    • $f_2: s_2 \Rightarrow s_3$
    • $(f_1 \circ f_2): s_1 \Rightarrow s_3$
Finite state morphology

- FSTs are invertible: what works for analysis also works for generation
- Doing away with rule ordering
  - Two-level morphology (Koskenniemi)
    - Surface and lexical levels related directly by an intersection of parallel FSTs, one for each rule
  - Composition of rule FSTs (Karttunen et al.)
    - $f_1: s_1 \Rightarrow s_2$
    - $f_2: s_2 \Rightarrow s_3$
    - $(f_1 \cdot o. f_2): s_1 \Rightarrow s_3$
    - The order of the rules is preserved in the composed FST that results
Finite state morphology

STEM

\[+s: [SING] \]

\[+': [SING, POSS] \]

\[': [PLUR, POSS] \]

\[[: [PLUR] \]

\[3\]

\[4\]
Finite state morphology

• For a given syntactic category in a language, a single FST consisting of a composed cascade of FSTs
Finite state morphology

• For a given syntactic category in a language, a single FST consisting of a composed cascade of FSTs
  - FST representing morphotactics
Finite state morphology
Finite state morphology

- For a given syntactic category in a language, a single FST consisting of a composed cascade of FSTs
Finite state morphology

- For a given syntactic category in a language, a single FST consisting of a composed cascade of FSTs
  - FST representing morphotactics
Finite state morphology

• For a given syntactic category in a language, a single FST consisting of a composed cascade of FSTs
  - FST representing morphotactics
  - FST representing possible stems
For a given syntactic category in a language, a single FST consisting of a composed cascade of FSTs
- FST representing morphotactics
- FST representing possible stems
- Ordered FSTs representing alternation rules (allomorphy, phonological rules, graphological rules)
Finite state morphology

- Morphotactics
  - Prefixes
  - Stem
  - Suffixes

LEXICAL

- Allomorphy
  - .O.
  - .O.

- Phonology/Orthography
  - .O.
  - .O.

SURFACE
Finite state morphology and non-concatenative morphology
Finite state morphology and non-concatenative morphology

- Finite state morphology inherently biased to view morphemes as sequences of characters and words as sequences of morphemes
Finite state morphology and non-concatenative morphology

- Finite state morphology inherently biased to view morphemes as sequences of characters and words as sequences of morphemes
- **Non-concatenative morphology**: infixation, circumfixation, base modification, reduplication, templatic morphology (interdigitation, intercalation, transfixation, root-pattern morphology)
Tigrinya
Tigrinya

- Seriously under-resourced
Tigrinya

- Seriously under-resourced
- Written in the Ge’ez script
Tigrinya

- Seriously under-resourced
- Written in the Ge’ez script
  - Syllabic
Tigrinya

- Seriously under-resourced
- Written in the Ge’ez script
  - Syllabic
  - Does not represent the (mostly) epenthetic vowel ɨ
Tigrinya

- Seriously under-resourced
- Written in the Ge’ez script
  - Syllabic
  - Does not represent the (mostly) epenthetic vowel ɨ
  - Does not represent gemination
Tigrinya imperfective verb morphotactics
Tigrinya imperfective verb morphotactics

STEM
Tigrinya imperfective verb morphotactics
Tigrinya imperfective verb morphotactics
Tigrinya imperfective verb morphotactics
Tigrinya imperfective verb morphotactics
Tigrinya imperfective verb morphotactics

SBJ1   STEM   (SBJ2)   [ (OBLQ) OBJ ]
ROOT
DERIVATION
Tigrinya imperfective verb morphotactics

\[
\text{(NEG1) SBJ1 STEM (SBJ2) [ (OBLQ) OBJ ] (NEG2)}
\]

\[
\text{ROOT}
\]

\[
\text{DERIVATION}
\]
Tigrinya imperfective verb morphotactics
Tigrinya imperfective verb morphotactics
Tigrinya imperfective verb morphotactics
Tigrinya imperfective verb morphotactics
Tigrinya imperfective
Tigrinya imperfective
Tigrinya imperfective

ኣብዘይትራኸባሉን
‘also where (=at which) you (plur. fem.) don’t meet’
Tigrinya imperfective

abzɛytraxɛbalun

ኣብዘይትራ nuesablu

‘also where (=at which) you (plur. fem.) don’t meet’
Tigrinya imperfective

abzɛytir_ amodxɛbalun
abzɛytraxɛbalun

ኣብዘይትራኸባሉን
abzɛytir_ amodxɛbalun
abzɛytraxɛbalun

��ትልይFaladvert
‘also where (=at which) you (plur. fem.) don’t meet’
also where (=at which) you (plur. fem.) don’t meet’
Tigrinya imperfective

STEM

\textit{ab\,zi\,ay\,ti\,r\_ak\_eb\,a\,l\,u\,n}\n
\textit{abz\_eytir\_ax\_ebalun} \quad \textit{abz\_eytrax\_ebalun}

‘also where (=at which) you (plur. fem.) don’t meet’
Tigrinya imperfective

DERIVATION: C₁aC₂εC₃

ROOT: r k b

STEM

ab zi ay ti r_akeb alun

abzεytir_axεbalun
abzεytraxεbalun

‘also where (=at which) you (plur. fem.) don’t meet’
Tigrinya imperfective

DERIVATION: $C_1_aC_2\varepsilon C_3$

ROOT: $r\ k\ b$

STEM

ab  zi  ay  ti  $r\_ak\varepsilon b$  a  l  u  n

$abz\varepsilon ytir\_ax\varepsilon balun$
$abz\varepsilon yt\varepsilon rax\varepsilon balun$

‘also where (=at which) you (plur. fem.) don’t meet’
Tigrinya imperfective

STEM

ab     zi     ay     ti         r_akεb     a    l    u    n

abzεytir axiombalun
abzεytraxεbalun

‘also where (=at which) you (plur. fem.) don’t meet’
Tigrinya imperfective

DERIVATION: \( C_1 \_aC_2 \_\varepsilon C_3 \) reciprocal

ROOT: \( r \_k \_b \) ‘find’

STEM

SUBJECT: 2 plur. fem.

\( ab \_zi \_ay \_ti \_r\_ak\_\varepsilon b \_a \_l \_u \_n \)

\( ab\varepsilon ytir\_a\varepsilon balun \)

\( ab\varepsilon yt\varepsilon trax\varepsilon balun \)

‘also where (=at which) you (plur. fem.) don’t meet’
Tigrinya imperfective

DERIVATION: $C_1_aC_2C_3$ reciprocal

ROOT: $r \ k \ b$ ‘find’

STEM

SUBJECT: 2 plur. fem.

OBJECT: 3 sing. masc. oblq.

abzəytiɾ_əxɛbalun
abzəytraxɛbalun

‘also where (=at which) you (plur. fem.) don’t meet’
Tigrinya imperfective

DERIVATION: $C_1\_aC_2\_C_3$

ROOT: $r\_k\_b$ ['find']

STEM

NEGATIVE

SUBJECT: 2 plur. fem.

OBJECT: 3 sing. masc. oblq.

‘also where (=at which) you (plur. fem.) don’t meet’
Tigrinya imperfective

DERIVATION: C₁ₐC₂εC₃ reciprocal

ROOT: \( r \ k \ b \) ‘find’

RELATIVE

NEGATIVE

STEM

SUBJECT: 2 plur. fem.

OBJECT: 3 sing. masc. oblq.

\( abzɛytr_ir_\_axɛbalun \)

\( abzɛytraxɛbalun \)

‘also where (=at which) you (plur. fem.) don’t meet’
Tigrinya imperfective

DERIVATION: $C_1_aC_2\varepsilon C_3$ reciprocal

ROOT: $r\quad k\quad b$ ‘find’

STEM: $r_{\_ak\varepsilon b}$

SUBJECT: 2 plur. fem.

OBJECT: 3 sing. masc. oblq.

PREPOSITION: $ab$- ‘at, in’

RELATIVE

NEGATIVE

abžeytir\_ax€balun
abżeytrax€balun

‘also where (=at which) you (plur. fem.) don’t meet’
Tigrinya imperfective

DERIVATION: $C_1 \alpha C_2 \varepsilon C_3$

ROOT: $r$ $k$ $b$ ‘find’

RELATIVE

NEGATIVE

STEM

SUBJECT: 2 plur. fem.

OBJECT: 3 sing. masc. oblq.

COORD CONJ: -n ‘and, also’

PREPOSITION: ab- ‘at, in’

abzęýtir_ąxębalun
abzęytraxębalun

‘also where (=at which) you (plur. fem.) don’t meet’
Tigrinya verb template morphology

<table>
<thead>
<tr>
<th></th>
<th>simple</th>
<th>pass/refl</th>
<th>caus</th>
<th>freqv</th>
<th>recip1</th>
<th>caus-recp1</th>
<th>recip2</th>
<th>caus-recp2</th>
</tr>
</thead>
<tbody>
<tr>
<td>perf</td>
<td>felη</td>
<td>tefe(e)l</td>
<td>afeη</td>
<td>felalet</td>
<td>tefeal</td>
<td>af_alet</td>
<td>tefeale</td>
<td>af_alet</td>
</tr>
<tr>
<td>imprf</td>
<td>fel (_i)l</td>
<td>fil _η</td>
<td>afe(_i)l</td>
<td>felal</td>
<td>felel</td>
<td>f_lel</td>
<td>tefeal</td>
<td>af_al</td>
</tr>
<tr>
<td>jus/impv</td>
<td>fel</td>
<td>tefel</td>
<td>afe</td>
<td>felal</td>
<td>tefelal</td>
<td>af_al</td>
<td>tefeal</td>
<td>af_al</td>
</tr>
<tr>
<td>ger</td>
<td>felηt</td>
<td>tefe(l)ηt</td>
<td>afeηt</td>
<td>felal</td>
<td>tefeal</td>
<td>af_alet</td>
<td>tefeale</td>
<td>af_alet</td>
</tr>
</tbody>
</table>

13
More on Tigrinya verbs
More on Tigrinya verbs

- Eight root categories
More on Tigrinya verbs

• Eight root categories
• Many phonological alternation rules
More on Tigrinya verbs

• Eight root categories
• Many phonological alternation rules
• Ambiguity
More on Tigrinya verbs

- Eight root categories
- Many phonological alternation rules
- Ambiguity
- Long-distance dependencies
More on Tigrinya verbs

- Eight root categories
- Many phonological alternation rules
- Ambiguity
- Long-distance dependencies
- A Tigrinya root can appear in well over 100,000 wordforms
Other approaches to Semitic template morphology
Other approaches to Semitic template morphology

- Multiple tapes: a transition relates a single surface character to multiple lexical characters (root, template) (Kiraz)
Other approaches to Semitic template morphology

• Multiple tapes: a transition relates a single surface character to multiple lexical characters (root, template) (Kiraz)

• Compile and replace: convert each combination of root and template ($rkb + C_1iC_2\varepsilon C_3$) into a separate FST, using the result as the appropriate stem (Beesley and Karttunen)
Weighted FSTs (Mohri)
Weighted FSTs (Mohri)

• Motivation
Weighted FSTs (Mohri)

• Motivation
  - For some applications, particular transitions are more or less preferred
Weighted FSTs (Mohri)

• Motivation
  - For some applications, particular transitions are more or less preferred
  - Paths through an FSA/FST need to be ranked
Weighted FSTs (Mohri)

• Motivation
  - For some applications, particular transitions are more or less preferred
  - Paths through an FSA/FST need to be ranked
• Semirings
Weighted FSTs (Mohri)

• **Motivation**
  - For some applications, particular transitions are more or less preferred
  - Paths through an FSA/FST need to be ranked

• **Semirings**
  - Algebraic structure with “addition” and “multiplication” operations, identity elements for each; addition identity element is annihilator for multiplication; multiplication distributes over addition
Weighted FSTs (Mohri)

• Motivation
  - For some applications, particular transitions are more or less preferred
  - Paths through an FSA/FST need to be ranked

• Semirings
  - Algebraic structure with “addition” and “multiplication” operations, identity elements for each; addition identity element is annihilator for multiplication; multiplication distributes over addition
  - Example: probabilities (“multiplication” is multiplication, “addition” is addition)
Weighted FSTs (Mohri)
Weighted FSTs (Mohri)

• Transitions are weighted with elements of a semiring
Weighted FSTs (Mohri)

• Transitions are weighted with elements of a semiring
• Most of the properties of FSTs hold
Weighted FSTs (Mohri)

- Transitions are weighted with elements of a semiring
- Most of the properties of FSTs hold
- Cumulative weight maintained for each path through the FSA/FST by “multiplying” the weights on the transitions
Weighted FSTs (Mohri)

- Transitions are weighted with elements of a semiring.
- Most of the properties of FSTs hold.
- Cumulative weight maintained for each path through the FSA/FST by “multiplying” the weights on the transitions.
- If there are multiple paths, an overall weight is assigned by “adding” the weights for each path.
Feature structures and unification
Feature structures and unification

- **Feature structures**: sets of attribute-value pairs; values are either atomic properties (FALSE, FEMININE) or feature structures
Feature structures and unification

- **Feature structures**: sets of attribute-value pairs; values are either atomic properties (FALSE, FEMININE) or feature structures
- *gezay* ‘my house’:
  
  \[
  \begin{array}{ll}
  \text{lex} & = \text{geza} \\
  \text{num} & = \text{sing} \\
  \text{poss} & = [\text{pers} = 1, \text{num} = \text{sing}] \\
  \end{array}
  \]
Feature structures and unification

- **Feature structures**: sets of attribute-value pairs; values are either atomic properties (FALSE, FEMININE) or feature structures
- **gezay ‘my house’**: 
  \[\text{[lex=geza, num=sing, poss=[pers=1,num=sing]]}\]
- **Operation of feature structures**: unification
Feature structures and unification

- **Feature structures**: sets of attribute-value pairs; values are either atomic properties (FALSE, FEMININE) or feature structures
- *gezay* ‘my house’:
  \[\text{[lex=geza, num=sing, poss=[pers=1, num=sing]]}\]
- Operation of feature structures: **unification**
  - Two FSs unify if their attribute-value pairs are compatible
Feature structures and unification

• **Feature structures**: sets of attribute-value pairs; values are either atomic properties (FALSE, FEMININE) or feature structures

• *gezay* ‘my house’:
  \[
  \text{[}\text{lex=geza, num=sing, poss=[pers=1,num=sing]}\text{]}
  \]

• Operation of feature structures: **unification**
  - Two FSs unify if their attribute-value pairs are compatible
  - Resulting unification combines features of the two FSs
Feature structures and unification

- **Feature structures**: sets of attribute-value pairs; values are either atomic properties (FALSE, FEMININE) or feature structures

- *gezay* ‘my house’:
  
  \[ \text{lex=geza, num=sing, poss=[pers=1, num=sing]} \]

- **Operation of feature structures**: *unification*
  
  - Two FSs unify if their attribute-value pairs are compatible
  - Resulting unification combines features of the two FSs
  - \[ \text{lex=geza, num=sing} \] and \[ \text{poss=[pers=1, num=sing]} \]
    
    unify to yield \[ \text{lex=geza, num=sing, poss=[pers=1, num=sing]} \]
Feature structures and unification

- **Feature structures**: sets of attribute-value pairs; values are either atomic properties (FALSE, FEMININE) or feature structures

- **gezay 'my house'**:  
  \[\text{lex=geza, num=sing, poss=[pers=1,num=sing]}\]

- **Operation of feature structures**: unification
  - Two FSs unify if their attribute-value pairs are compatible
  - Resulting unification combines features of the two FSs
  - \[\text{lex=geza, num=sing}\] and \[\text{poss=[pers=1, num=sing]}\] unify to yield  
    \[\text{lex=geza, num=sing, poss=[pers=1, num=sing]}\]
  - TOP unifies with anything
FSTs weighted with feature structure sets (Amtrup)
FSTs weighted with feature structure sets (Amtrup)

- Sets of FSs constitute a semiring: pairwise unification as “multiplication” and set union as “addition”
FSTs weighted with feature structure sets (Amtrup)

• Sets of FSs constitute a semiring: pairwise unification as “multiplication” and set union as “addition”
• FSTs weighted with FS sets:
FSTs weighted with feature structure sets (Amtrup)

- Sets of FSs constitute a semiring: pairwise unification as “multiplication” and set union as “addition”
- FSTs weighted with FS sets:
  - Traversing path yields FS set in addition to output string, the result of the repeated unification of FS sets on the transitions in the network, starting with an initial FS set
FSTs weighted with feature structure sets (Amtrup)

- Sets of FSs constitute a semiring: pairwise unification as “multiplication” and set union as “addition”
- FSTs weighted with FS sets:
  - Traversing path yields FS set in addition to output string, the result of the repeated unification of FS sets on the transitions in the network, starting with an initial FS set
  - Path fails not only if the input character fails to match the input character on the arc, but also if the accumulated FS set fails to unify with the FS set on the arc
FSTs weighted with feature structure sets (Amtrup)

• Sets of FSs constitute a semiring: pairwise unification as “multiplication” and set union as “addition”

• FSTs weighted with FS sets:
  - Traversing path yields FS set in addition to output string, the result of the repeated unification of FS sets on the transitions in the network, starting with an initial FS set
  - Path fails not only if the input character fails to match the input character on the arc, but also if the accumulated FS set fails to unify with the FS set on the arc

• Advantages
FSTs weighted with feature structure sets (Amtrup)

- Sets of FSs constitute a semiring: pairwise unification as “multiplication” and set union as “addition”
- FSTs weighted with FS sets:
  - Traversing path yields FS set in addition to output string, the result of the repeated unification of FS sets on the transitions in the network, starting with an initial FS set
  - Path fails not only if the input character fails to match the input character on the arc, but also if the accumulated FS set fails to unify with the FS set on the arc
- Advantages
  - FST outputs feature structures
FSTs weighted with feature structure sets (Amtrup)

- Sets of FSs constitute a semiring: pairwise unification as "multiplication" and set union as "addition"

- FSTs weighted with FS sets:
  - Traversing path yields FS set in addition to output string, the result of the repeated unification of FS sets on the transitions in the network, starting with an initial FS set
  - Path fails not only if the input character fails to match the input character on the arc, but also if the accumulated FS set fails to unify with the FS set on the arc

- Advantages
  - FST outputs feature structures
  - Long-distance dependencies handled efficiently
FSTs weighted with feature structure sets (Amtrup)

- Sets of FSs constitute a semiring: pairwise unification as “multiplication” and set union as “addition”
- FSTs weighted with FS sets:
  - Traversing path yields FS set in addition to output string, the result of the repeated unification of FS sets on the transitions in the network, starting with an initial FS set
  - Path fails not only if the input character fails to match the input character on the arc, but also if the accumulated FS set fails to unify with the FS set on the arc
- Advantages
  - FST outputs feature structures
  - Long-distance dependencies handled efficiently
FSTs weighted with feature structure sets (Amtrup)

- Sets of FSs constitute a semiring: pairwise unification as “multiplication” and set union as “addition”
- FSTs weighted with FS sets:
  - Traversing path yields FS set in addition to output string, the result of the repeated unification of FS sets on the transitions in the network, starting with an initial FS set
  - Path fails not only if the input character fails to match the input character on the arc, but also if the accumulated FS set fails to unify with the FS set on the arc
- Advantages
  - FST outputs feature structures
  - Long-distance dependencies handled efficiently
Weighted FSTs applied to Tigrinya verbs: long-distance dependencies
Weighted FSTs applied to Tigrinya verb stems
Weighted FSTs applied to Tigrinya verb stems

- Handle the root characters with the normal FST machinery
Weighted FSTs applied to Tigrinya verb stems

- Handle the root characters with the normal FST machinery
- Handle the template characters with grammatical features
Weighted FSTs applied to Tigrinya verb stems

- Handle the root characters with the normal FST machinery
- Handle the template characters with grammatical features
- $q_{\varepsilon\varepsilona\varepsilon l} \leftrightarrow q_{\varepsilon l}, \text{[der= [+ps, -tr, -rc, +it]]}$
Weighted FSTs applied to Tigrinya verb stems
Weighted FSTs applied to Tigrinya verb stems

- Stems for CC_C root class
Tigrinya imperfective FST

flṭ; $[\text{der=[+ps,-tr,-it,-rc],}$
$sbj=[+2p,+plr,-fem], +neg]}$

`...'aytifik translators...

Prefixes

Stem (Root+Pattern)

Suffixes

Allomorphy

Phonology

Orthography

λ anlamı (‘aytfíלק tun)
Tigrinya imperfective FST
Tigrinya imperfective FST

• 8757 states, 49137 transitions
Tigrinya imperfective FST

- 8757 states, 49137 transitions
- Evaluation
Tigrinya imperfective FST

- 8757 states, 49137 transitions
- Evaluation
  - Analysis
Tigrinya imperfective FST

• 8757 states, 49137 transitions
• Evaluation
  - Analysis
    • 250 randomly selected words from a database of 227,984 distinct Tigrinya wordforms (Biniam)
Tigrinya imperfective FST

- 8757 states, 49137 transitions
- Evaluation
  - Analysis
    - 250 randomly selected words from a database of 227,984 distinct Tigrinya wordforms (Biniam)
    - All 182 unambiguously imperfective were analyzed correctly
Tigrinya imperfective FST

- 8757 states, 49137 transitions
- Evaluation
  - Analysis
    - 250 randomly selected words from a database of 227,984 distinct Tigrinya wordforms (Biniam)
    - All 182 unambiguously imperfective were analyzed correctly
    - 51 non-imperfective verbs either rejected or analyzed as forms of non-existent roots
Tigrinya imperfective FST

- 8757 states, 49137 transitions
- Evaluation
  - Analysis
    - 250 randomly selected words from a database of 227,984 distinct Tigrinya wordforms (Biniam)
    - All 182 unambiguously imperfective were analyzed correctly
    - 51 non-imperfective verbs either rejected or analyzed as forms of non-existent roots
    - Produces as many as seven different analyses, for roots
Tigrinya imperfective FST

- 8757 states, 49137 transitions
- Evaluation
  - Analysis
    - 250 randomly selected words from a database of 227,984 distinct Tigrinya wordforms (Biniam)
    - All 182 unambiguously imperfective were analyzed correctly
    - 51 non-imperfective verbs either rejected or analyzed as forms of non-existent roots
    - Produces as many as seven different analyses, for roots
  - Generation
Tigrinya imperfective FST

- 8757 states, 49137 transitions
- Evaluation
  - Analysis
    - 250 randomly selected words from a database of 227,984 distinct Tigrinya wordforms (Biniam)
    - All 182 unambiguously imperfective were analyzed correctly
    - 51 non-imperfective verbs either rejected or analyzed as forms of non-existent roots
    - Produces as many as seven different analyses, for roots
  - Generation
    - 272 tests: roots from all categories, all possible derivational patterns, various combinations of affixes
Tigrinya imperfective FST

- 8757 states, 49137 transitions
- Evaluation
  - Analysis
    - 250 randomly selected words from a database of 227,984 distinct Tigrinya wordforms (Biniam)
    - All 182 unambiguously imperfective were analyzed correctly
    - 51 non-imperfective verbs either rejected or analyzed as forms of non-existent roots
    - Produces as many as seven different analyses, for roots
  - Generation
    - 272 tests: roots from all categories, all possible derivational patterns, various combinations of affixes
    - All generated correctly
Conclusions
Conclusions

• Tigrinya morphology is complex
Conclusions

- Tigrinya morphology is complex
- FSTs with FS weights makes it much simpler
Conclusions

• Tigrinya morphology is complex
• FSTs with FS weights makes it much simpler
• Future work: sharing of knowledge also facilitates morphological translation between similar languages
አለ thermostat!  
Thank you!