A computational approach to Yorùbá morphology

Raphael Finkel
Computer Science Department
University of Kentucky,
USA

Ọdetúnjí A. Ọdejobí
Cork Constraint Computation Center (4C)
Computer Science Department
University College Cork
Cork, Ireland

SUPPORTED BY US National Science Foundation Grants IIS-0097278 and IIS-0325063
and by the University of Kentucky Center for Computational Science

SUPPORTED BY Science Foundation Ireland Grant 05/IN/I886 and Marie Curie Grant MTKD-CT-2006-042563

31 March 2009
Yoruba verb morphology

In this presentation

- Explain the output of our program for Yoruba verb morphology
  
  /home/odetunji/Desktop/ConferenceSlides/yoruba.utf8.html

- Discuss how we developed the program

- Discuss the significance of our efforts

- State our ongoing efforts
Yorùbá in Brief

• Edikiri language in the Niger-Congo family spoken widely in southwestern Nigeria (ISO: yor)
• Many dialects, with a standard form (SY) for communication and education
• 3 tones: High(H), Medium(M), Low(L)
• 2 tonal contours: falling (HL) and rising (LH)
• Simple verb morphology: Only one conjugation

• The verb morphology is documented.
Our goals

To generate verb forms for SY

(i) realise all 160 combinations of morphosyntactic properties

- **Tense**: present, continuous, past, future
- **Polarity**: positive, negative
- **Person**: 1, 2Older, 3Older, 2Notolder, 3NotOlder
- **Number**: singular, plural
- **Strength**: normal, emphatic

(ii) provide a computational description of SY verb formation
The KATR formalism

- Based on DATR, a formalism for representing lexical knowledge by default-inheritance hierarchies (Evans & Gazdar, 1989).

- Queries (such as 1 pl past) are directed to nodes that contain rules that either answer the queries or direct them to further nodes.
Generating Queries in KATR

We declare variables to represent morphosyntactic properties

1) #vars $tense: present past continuous future .
2) #vars $polarity: positive negative .
3) #vars $person: 1 2Older 3Older 2NotOlder 3NotOlder .
4) #vars $number: sg pl .
5) #vars $strength: normal emphatic .
Generating multiple queries

#show <$strength :: $polarity :: $tense :: $person :: $number > .

- This "show" line generates 160 queries such as:
  - <normal negative past 3Older sg>
  - <emphatic negative continuous 3Older pl>

- These queries are directed to all leaf nodes, such as the "Take" node. (Node names always start with upper-case letters)
The "Take" node

Take:

1. <stem> = m un \( \acute{\cdot} \) % tone marks always follow vowels
2. {} = Verb

- The order of rules is not significant.
- The query <emphatic negative continuous 3Older pl> only matches Rule 2, which is completely unconstrained.
- Rule 2 directs the query to the "Verb" node.
The "Verb" node

Verb:

1 \{\} = Person Negator1 Tense Negator2 , "<stem>" Ending
2 \{continuous negative\} = <present negative>

This query:

<emphatic negative continuous 3Older pl>

matches both rules. KATR chooses the more constraining rule (Panini's principle), that is, Rule 2.

Rule 2 converts the query to

<present negative emphatic 3Older pl>

and directs it again to the "Verb" node.
The "Verb" node, modified query

Verb:
1 \{\} = Person Negator1 Tense Negator2, "<stem>" Ending
2 \{continuous negative\} = <present negative>

This modified query:

<present negative emphatic 3Older pl>
matches only Rule 1, which

- Represents our analysis of SY, which identifies 6 slots.
- Combines the results for each slot into a single result
  - The results of sending the query to five different nodes.
  - The surface form ",," which we use to create word boundaries.
  - The result of sending the new query "<stem>" to the starting leaf node "Take", which returns the surface form “m un´”
The "Person" node

Person:
1 \{3Older positive !future\} = w ϙ n´
2 \{3Older\} = w ϙ n
3 \{3NotOlder\} = o´
4 \{3NotOlder negative sg\} =
5 \{3NotOlder future\} = y i´
6 \{3NotOlder pl ++\} = <3Older>
... % omitting many other rules

This query:
<present negative emphatic 3Older pl></p>
only matches Rule 2, generating the answer “w ϙ n”.
The "Negator1" node

Negator1:
1 \{negative\} = , (k) o ` \\
2 \{negative 3NotOlder sg\} = k o ` \\
3 \} = \\

This query:
<present negative emphatic 3Older pl>

matches Rules 1 and 3. KATR chooses Rule 1, generating the answer “, (k) o `”.
The "Tense" node

Tense: % polarity, tense
1 {} =
2 {past} = , t i
3 {continuous positive} = , n´
4 {future positive} = , o^ 
5 {future 1 sg positive} = , a` 
6 {future 3NotOlder positive} = <future 3Older positive>

This query:
<present negative emphatic 3Older pl>

matches Rule 1, generating an empty (but valid!) output.
The "Negator2" node

Negator2: % polarity, tense
1 {future negative} = , n i ´
2 {past negative} = ´ i `.
3 {} =

This query:
<present negative emphatic 3Older pl>

Matches only Rule 3, which generates an empty output.
The "Ending" node

Ending:
1 \{\} =
2 \{emphatic\} = ↓

This query:
<present negative emphatic 3Older pl>

Matches both rules; KATR chooses Rule 2, which generates ↓, which is a jer for post-processing.
Postprocessing

The "Verb" node assembles all the results into this surface form:

\[ w \quad n, \ (k) \ o \ ` , \ m \ u n \ ` \]

This surface form is now treated by postprocessing rules.

1) \#sandhi $vowel \downarrow \Rightarrow $1 $1 `.
2) \#sandhi $vowel $tone \downarrow \Rightarrow $1 $2 $1 .
3) \#sandhi un $tone \Rightarrow u $1 n . \% spelling
4) \%(others omitted)

Rules 1 and 2 remove the \downarrow jer. In this case, Rule 2 applies, giving us:

\[ w \quad n, \ (k) \ o \ ` , \ m \ u n \ ` \ un \]
Then Rule 3 applies, giving us

\[ \text{won} , (k) \ o \ ` , m \ u \ ` n \ un \]

When we compress spaces out and replace comma with space, we get:

\[ \text{won} (k)\ò \mí\n\]

which is the correct surface form for

Take:<emphatic negative continuous 3Older pl>

“They (older) are certainly not taking (that object)”
Implementation

1. A Perl script converts the KATR theory into
   - yoruba.katr.pro: a Prolog representation of the theory
   - yoruba.sandhi.pl: a Perl script for post-processing

2. A Prolog interpreter computes the results of all queries generated by "show" directed to all leaf nodes in the KATR theory.

3. The Perl post-processing script applies the Sandhi and other post-processing rules.

4. We then either generate textual output for direct viewing or HTML output for a browser.

The KATR theory implementation for Yoruba is available at http://www.cs.uky.edu/~raphael/KATR.html
Applications

- **Linguistics**: Theoretical studies of SY
- **Pedagogy**: Describing SY verbs to students
- **Learning**: Facilitating tool for teaching SY
- **Technology**: Developing software products such as spelling and grammar checkers
KATR instead of DATR

- KATR is fast, so turn-around time is very short.

- KATR allows sets in addition to paths on the left-hand side, so it is easy to ignore irrelevant morphosyntactic properties.

- KATR lets us specify post-processing directly instead of embedding it in the default-inheritance hierarchy.
Contributions

- Description of slots in SY verb morphology
  - Six slots identified

- Complete specification of the realizations of those slots

- A simple use of *jers* to deal with the tone Sandhi of the emphatic suffix.
On going efforts

- **Evaluation**: Subject out program to further evaluation throughout working with Yoruba linguists and phonologist
- **Expansion**: Expand the rule for similar African tone languages
- **Exploration**: Explore the generality of our approach and the possibility for developing generic morphological rules
HELP!!
Suggestions?
Education?
Questions?