HIGH SCHOOL EDITION
Journal of Student Research
Volume 12 Issue 4 (2023)

Coding Local Weight Sensitivity in Wergaia

Kuo-Chiao Lin[1] and Kyle Ho[#]

[1]Kang Chico International School, Taiwan
[#]Advisor

ABSTRACT

Wergaia, an extinct language in Australian Wimmera region, exemplifies a generalized trochee language wherein heavy syllables generally do not bear stress, except when in final position in an odd-parity word to preserve a regular rhythmic alternation. This paper reviews the Wergaia stress pattern and provides a program in Python to model its local weight sensitivity in metrical parsing.

A Geo-Historical Overview of Wergaia

In the quiet expanse of the Australian Wimmera region, a language once whispered in the rustling leaves of mallee eucalypt bushlands now stirs. Wergaia, an extinct Aboriginal tongue, reveals its nuanced tapestry, a symphony of dialects: Wudjubalug, Djadjala, Buibadjali, and Biwadjali (Clark 1990). These dialects, like the subtle shifts in wind through the branches, tell the stories of a people, the Maligundidj or Wergaia, bearers of the land's name, guardians of its secrets. Intricately woven into the fabric of the Kulinic branch of the Pama-Nyungan language family, Wergaia's roots extend deep into the soil of the Wemba Wemba language, connecting generations with the threads of their heritage (Dixon 2002). In the gentle embrace of the year 2021, a language revival project began at the Wotjobaluk Knowledge Place, a haven that took root in the heart of Dimboola the year prior, in order to bring these dialects to new life (Kelso 2021).

Local Weight Sensitivity in Wergaia

In Wergaia, the stress pattern, as (1) shows, exhibits a specific structure where the primary stress consistently falls on the initial syllable of a word, while secondary stresses occur on the remaining odd-numbered syllables within that word. Notably, final syllables in Wergaia generally do not receive stress, as shown in (1a-b). However, there is an interesting exception: a final odd-numbered syllable will receive stress only if it is considered heavy, as in (1c). This results in a trochee stress pattern that can be characterized as a canonical generalized trochee stress pattern. This pattern has been discussed in linguistic literature by scholars such as Prince (1980), Kager (1992), and Hayes (1995). It is essential to understand that Wergaia restricts vowel length to the initial syllable position (Hercus 1986: 81). Consequently, only closed syllables within words with odd syllable counts can carry final stress. In other words, within Wergaia's predominantly quantity-insensitive trochaic stress system, weight sensitivity comes into play in a limited manner. This weight sensitivity is employed in Wergaia to maintain an alternating rhythm (Hercus 1986).

(1) Wergaia stress (adapted from Pruitt 2012: 81)
a. Words with an even number of syllables:

ˈgaba	'to chase'	(ˈLL)
ˈbaɟig	'stone tomahawk'	(ˈLH)
ˈwinagˌɟera	'to leave one another'	(ˈLH)(ˌLL)

ˈwiɾim ˌbuliɲ ‘spider’ (ˈLH)(ˌLH)

b. Odd-parity words with a final light syllable:

ˈdaguŋga ‘to punch someone’ (ˈLH)L

ˈdelguna ‘to cure’ (ˈHL)L

c. Odd-parity words with a final heavy syllable:

ˈbunaˌɖug ‘broad-leaved mallee’ (ˈLL)(ˌH)

ˈgeɟauˌwil ‘he let me in’ (ˈLH)(ˌH)

Coding Weight Sensitivity in Wergagia

The algorithm that outlines the steps the Python code for analyzing Wergaia’s syllable structure and stress pattern is shown in (2).

(2) *Algorithm for Wergaia stress*

a. Define the lists of vowels and consonants used in the Wergaia language.

b. Define functions for checking character types:

i. is_vowel(a): Returns true if the character is a vowel.

ii. is_consonant(b): Returns true if the character is a consonant.

iii. is_binded_consonant(bcd): Returns true if the character(s) form complex consonants.

c. Define a function to check whether a syllable is “heavy”:

i. is_heavy(syllable): Returns true if the syllable is considered heavy.

d. Define a function to break a given input string into individual syllables based on Wergaia’s phonological rules:

i. to_syllables(string): Processes the input string and returns a list of syllables.

e. Iterate through the list of syllable indices and adjust syllables based on the presence of special consonant combinations to ensure they adhere to Wergaia phonotactics.

i. Define a function to determine the stress pattern for each syllable:

- to_syllables_stress(syllables): Assigns stress patterns to syllables according to the rules:
  - Primary stress to the first syllable (if it's the first syllable in the word).
  - Secondary stress to even-numbered syllables if they are not the last syllable and not heavy.
  - Unstressed to all other syllables.

f. Define a function to print the syllables with their associated stress patterns:

i. print_syllables_stress(syllables_stress): Prints the word with stress markers:

- " ˈ " for primary stress.
- " ˌ " for secondary stress.
- No stress marker for unstressed syllables.

g. In the main() function:

i. Prompt the user to enter a word.

ii. Call the to_syllables(string) function to break the word into syllables.

iii. Call the to_syllables_stress(syllables) function to determine stress patterns for each syllable.

iv. Call the print_syllables_stress(syllables_stress) function to display the word with stress

markers.

This algorithm provides a step-by-step guide to how the Python code processes and analyzes Wergaia words, taking into account phonological rules and stress patterns. The resulting Python code according to the algorithm is given in (3).

(3) *Python code for Wergaia stress*

```
vowels = ['a', 'ɑ', 'æ', 'ɐ', 'ɑ̃',
          'e', 'ə', 'ɚ', 'ɵ',
          'ɛ', 'ɜ', 'ɝ', 'ɛ̃',
          'i', 'ɪ', 'ɨ', 'ɪ̈',
          'o', 'ɔ', 'œ', 'ɒ', 'ɔ̃',
          'ø',
          'u', 'ʊ', 'ʉ']
consonants = [['b', 'β', 'ɓ',
               'c', 'ç', 'ɕ',
               'd', 'ð', 'ɗ', 'ɖ',
               'f',
               'g', 'ɠ', 'ɢ',
               'h', 'ħ', 'ɦ', 'ɥ', 'ɧ', 'ʜ',
               'j', 'ɟ', 'ʄ',
               'k',
               'l', 'ɬ', 'ɭ', 'ɮ', 'ʟ', 'ɻ',
               'm', 'ɱ',
               'n', 'ŋ', 'ɲ', 'ɳ', 'ɴ',
               'p', 'ɸ',
               'q',
               'r', 'ɾ', 'ɹ', 'ʁ', 'ʀ', 'ɺ', 'ɽ',
               's', 'ʃ', 'ʂ',
               't', 'θ', 'ʈ',
               'v', 'ʌ', 'ʋ',
               'w', 'ɯ', 'ʍ',
               'x', 'χ',
               'y', 'ɣ', 'ʎ', 'ʏ', 'ʏ',
               'z', 'ʒ', 'ʐ', 'ʑ'],
               ['d͡ʒ',
                't͡ʃ', 't͡s']]

def is_vowel(a):
    return a in vowels

def is_consonant(b):
    return b in consonants[0]

def is_binded_consonant(bcd):
    return bcd in consonants[1]
```

```
def is_heavy(syllable):
    return (len(syllable) > 1 and is_consonant(syllable[len(syllable)-1])) or (len(syllable) > 3 and is_binded_consonant(syllable[len(syllable)-3:]))

def to_syllables(string):
    syllables = []
    syllable_indices = []
    for i in range(len(string)):
        if is_vowel(string[i]):
            if i == 0 or (i >= 1 and is_consonant(string[i-1])) or (i >= 3 and is_binded_consonant(string[i-3:i])):
                syllables += string[i]
                syllable_indices += [i]
            else:
                syllables[len(syllables)-1] += string[i]
    for i in range(len(syllable_indices)):
        index = syllable_indices[i]
        if index >= 3 and is_binded_consonant(string[index-3:index]):
            syllables[i] = string[index-3:index+1]
        elif index >= 1 and is_consonant(string[index-1]):
            syllables[i] = string[index-1:index+1]
        if index + 3 < len(string) and is_binded_consonant(string[index+1:index+4]):
            if not (index + 4 in syllable_indices) or index + 4 >= len(string):
                syllables[i] = syllables[i] + string[index+1:index+4]
        elif index + 1 < len(string) and is_consonant(string[index+1]):
            if not (index + 2 in syllable_indices) or index + 2 >= len(string):
                syllables[i] = syllables[i] + string[index+1]
    return syllables

def to_syllables_stress(syllables):
    syllables_stress = []
    for syllable in syllables:
        syllables_stress += [[syllable,"undetermined"]]
    for i in range(len(syllables)):
        if i % 2 == 0:
            if i == 0:
                syllables_stress[i][1] = "primary"
            elif i != len(syllables) - 1 or is_heavy(syllables[i]):
                syllables_stress[i][1] = "secondary"
            else:
                syllables_stress[i][1] = "unstressed"
        else:
            syllables_stress[i][1] = "unstressed"
    return syllables_stress

def print_syllables_stress(syllables_stress):
    for i in range(len(syllables_stress)):
```

```
            match syllables_stress[i][1]:
                case 'primary':
                    print(" ˈ " + syllables_stress[i][0], end="")
                case 'secondary':
                    print(" ˌ " + syllables_stress[i][0], end="")
                case 'unstressed':
                    print(syllables_stress[i][0], end="")

def main():
    string = input("Please enter a word: ")
    syllables = to_syllables(string)
    syllables_stress = to_syllables_stress(syllables)
    print_syllables_stress(syllables_stress)

main()
```

Conclusion

This paper reviews the stress distribution in Wegaia and offers a Python program capable of modeling local weight sensitivity in this extinct language. Future programming agenda may include a wider spectrum of quantity-sensitive phenomena in both theoretical and crosslinguistic perspectives, such as trochaic shortening in Fijian (Dixon 1988) and schwa reduction in Piuma Paiwan (Shih 2019), both of which are quantity-sensitive. How to model these phenomena in Python awaits further programming attempts.

Acknowledgments

I would like to thank Kuo-Chiao Lin for introducing me to the field of linguistics and his unwavering support and guidance.

References

Clark, Ian (1990). Aboriginal languages and clans: an historical atlas of Western and Central Victoria, 1800-1900. Monash Publications in Geography 30.

Dixon, Robert M. W. 1988. A grammar of Boumaa Fijian. University of Chicago Press, Chicago.

Dixon, Robert M. W. *2002.* Australian languages: their nature and development. Cambridge University Press, Cambridge.

Hayes, Bruce. 1995. Metrical stress theory: principles and case studies. University of Chicago Press, Chicago.

Hercus, Louise. 1986. Victorian languages: a late survey. Australian National University, Canberra.

Kager, René. 1992. Shapes of the generalized trochee. *Proceedings of the West Coast Conference on Formal Linguistics* 11: 298–312.

Kelso, Andrew. 2021. Dimboola to 'revive' Wergaia language, in Victorian first. *ABC News*. Australian Broadcasting Corporation. (Retrieved 21 June 2021).

Prince, Alan. 1980. A metrical theory for Estonian quantity. *Linguistic Inquiry* 11: 511–562.

Pruitt, Kathryn. 2012. Stress in Harmonic Serialism. PhD dissertation, University of Massachusetts Amherst.

Shih, Shu-hao. 2019. The phonetics and phonology of non-moraic schwas: new evidence from Piuma Paiwan. *Proceedings of the 49th Annual Meeting of the North East Linguistic Society*, 137–148. Amherst, MA: GLSA.