Computational Complexity in Optional Syllabification of Yavapai

Wenyue Hua*

1 Introduction

In addition to substance in phonology, a number of researchers have argued that computation also matters in phonology (Jardine 2019, Chandlee 2014, Heinz 2018). Using the data in Yavapai (Yuman language), I show that other than an OT analysis focusing mainly on substance, a computational analysis is necessary for explaining the complex syllabification processes and the frequencies of optional surface representations (SRs) due to different syllabifications (Shaterian 1983). I will use computational complexity encoded in the subregular hierarchy as the main technical tool in the computational analysis. Our main hypothesis is that when both SRs are well-formed based on the syllable phonotactics, the one less complex to generate is more frequently attested. The paper shows that the syllabification pattern in Yavapai necessarily requires a computational motivation, which in turn shows that computational property is a crucial factor in phonological transformations.

The paper is organized as follows. Section 2 introduces relevant data points and their distributions of syllabification of Yavapai. Section 3 discusses the reason why Optimality Theory (OT), which focuses on the substance part of phonology, fails to provide a complete explanation of the phenomenon. Section 4 motivates a complexity explanation and explicates relevant concepts and definitions in subregular hierarchy to formally explain the data distribution. The last section concludes the paper.

2 Data Distribution

Yavapai is an Upland Yuman language. It was once spoken across much of north-central and western Arizona but is now mostly spoken on the Yavapai reservations at Fort McDowell, the Verde Valley and Prescott. The data of this paper consults the dissertation *Phonology and Dictionary of Yavapai* written by Shaterian (1983).

One special characteristic of Yavapai language is that most affixes in the language are monosegmental. Below are some examples:

(1)	/t͡ʃ-/	/nali/	→	/t͡ʃnaːliːti/
	CAUSATIVE	fall	→	drop
(2)	/k-/	/myala/	→	/kmyaːla/
	AGENTIVE	bread	→	baker
(3)	/ʔ/	/βyam/	→	/ʔβyamkm/
	first person	run	→	I am running

Due to this property and the fact that Yavapai is poor in syllabic or vocalic segments, this highly agglutinated language contains many consonant clusters in the underlying representations (URs). Below are examples with morphological analysis presented for each affix and the roots:

(4)	/ʔʔont͡ʃ/	/ŋjeːkkkθoː/
	the fire	tomorrow
	ʔ-ʔo-n-t͡ʃ	ŋ-jeːk-k-k-θ-oː
	NOUN-MARKER-fire-DEMONSTRATIVE-SUBJECT	TEMPORAL.SUBORDINATOR-dawn-SAME.SUBJECT-RELATIVIZER-CONTRASTIVE.MODAL-TEMPORAL

Because of Yavapai's rich morphology and simple root, the URs of words are easy to determine: they

*I would like to thank Adam Jardine for his insights and feedback. Additionally, thanks go out to Adam McCollum, Bruce Tesar and the audience at RULing 14 at Rutgers. All errors are my own.

are compositionally composed of morphemes and roots. Therefore we will adopt the URs directly from the dissertation without further justifying them in the rest of the paper.

2.1 Basic Syllabification Pattern

This language requires an onset for each syllable but disfavors complex onsets or complex codas, thereby almost all syllables are of the canonical syllable structure CV or CVC on SRs. One of the most important phonological processes is how to break consonant clusters in a syllable. Notice that since almost all monosegments in URs bear non-vacuous syntactic or semantic functions, deletion is never applied to avoid consonant clusters. Instead, gemination (Davis 2011, Ridouane 2010, etc), usually defined as a phonetic doubling of consonants, is the main process that creates syllables among consonant clusters. Below are some examples:

(5) /ʔsaː/ [ʔs̩.saː] eagle

(6) /ʔwa/ [ʔu.wa] rock

(7) /tki/ [təkˀ.ki] it hurts

The three simple data points above each contains one consonant cluster: /ʔs/, /ʔw/ and /tk/. To avoid a consonant cluster in a syllable, the two consonants constitute a new syllable: [ʔs̩], [ʔu] ([ʔw̩]) and [təkˀ] where the second consonants are syllabified or have a schwa epenthesized when the consonant is a stop, and become the nucleus of the created syllables. Since Yavapai requires every syllable to have an onset, the second syllables [s, w, k] are also the onset of the next syllables. Contrary to the examples above, consonant clusters are mostly tolerated when they can be syllabified into different syllables. Below are the examples:

(8)

[βətˀ.taβ.si]	snap shut	[kl̩.lum.ʔi]	limp from arthrits
[km̩.puːl.βa]	hump(back)	[sm̩.kʷiːr.βi]	jealous (man)

2.2 Optionality and Frequency

The description above demonstrates the general pattern of syllabification, realized by gemination of consonants, when there are only two consonants in the consonant cluster. However, the pattern grows more complicated when a consonant cluster becomes longer: more than one syllabifications are available. Below are some examples:

(9) /ʔmpat͡ʃa/ [ʔm̩.pa.t͡ʃa] ∼ [ʔm̩.məpˀ.pa.tʃa] brush, plant

(10) /smpuːrβi/ [sm̩.puːr.βi] ∼ [sm̩.məpˀ.puːr.βi] band for cradle

(11) /mltat/ [ml̩.tat] ∼ [ml̩.lətˀ.tat] barrel cactus

(12) /tʔruji/ [təʔˀ.ru.ji] ∼ [təʔˀ.ʔɾ̩.ru.ji] make hot

All examples above have two possible realizations of the surface forms for a string of segments $C_1C_2C_3V$: one contains three syllables in the form of $C_1C_2.C_2C_3.C_3V$, and the other contains two syllables in the form of $C_1C_2.C_3V$. Therefore the optionality is indeed the optionality of whether C_2 and C_3 form a syllable. This is reasonable since syllables of all SRs are well-formed.

It is important to notice that the long forms occur much more often than the short one. Based on Shaterian (1983, Appendix I), there are in total 95 different types and 105 occurrences of relevant consonant clusters recorded. The numbers and frequencies of the two forms are summarized and presented below.

(13)

form	number of occurrences	percentage
long form	76	72.4%
short form	29	17.6%

Based on the dissertation we consult, no recorded optionality has occurred when there are four or more consonants in a cluster. We do not have enough data points to make any sound generalization

here. Below are some examples:

(14) /kkβkoːβi/ [kəkˀ.kβ̩.βəkˀ.koː.βi] shield

(15) /sklpuji/ [səkˀ.kl̩.ləpˀ.pu.ji] hug

(16) /ʔtʃkŋe/ [ʔtʃ̩.kŋ̩.ŋe] hunter

(17) /ʔuːβa tʃmmjialβa/ [ʔuː.βa tʃ̩m.mi.ja.lβ̩.βa] cigar

2.3 Summary

This section introduced the data of syllabification in Yavapai. Syllabification which is realized by gemination of consonants is the main process that helps avoid consonant clusters from surfacing. Therefore, one task of a phonological analysis is to demonstrate how the syllabification phenomenon is generated. In addition, different SRs occur in free variantion when the underlying consonant clusters contain more than two consonants: long form and short form.

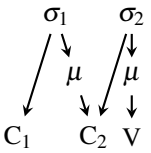
3 Data Analysis and Problems of Optimality Theory

This section discusses how and to what extent Optimality Theory can explain the phenomena and data presented above. In order to describe and then explain how the the syllabification works and how new syllables are formed, choosing a theory about syllable structure is crucial to the analysis of these processes. There are mainly two theories about syllable structure: CV theory and moraic theory.

There are mainly two arguments against using CV theory here: First, when C_2 is a stop, the repair of complex consonantal onset unnecessarily results in a syllable with a coda, while C_1 with a schwa epenthesized could have formed a canonical CV syllable. We could posit that this language prefers some syllables with a coda. But this is both theoretically undesirable and untrue for other data points in the language because dispreference against codas is typologically universal and there are many syllables without codas recorded. Second, the nucleus of a syllable in the language is not necessarily a vowel while many syllabic [+consonantal] segments can bear the role of the nucleus. In CV theory, V can freely dominate a [-consonantal] segment but syllabic consonants are still [+consonantal]. The dominance of V on these syllabic consonants is thereby problematic when no further theoretical claims are made. Therefore a CV skeleton theory is not suitable here to discuss the syllable structure in the language. We will adopt the moraic theory of syllable structure as an assumption in our discussion.

Since only segmental representations are given in the dissertation, we hypothesize the correspondence below between the segmental representation recorded in the dissertation and the moraic representation (Tranel 1991, Hayes 1989):

(18) $C_1C_2.C_2V$:



The syllabic C_2 in segmental representation which is the nucleus of the first syllable is dominated by a mora from the first syllable while the onset C_2 in segmental representation is dominated directly by the second syllable.

The main motivation under the syllabification process is a highly ranked markedness constraint that punishes a complex onset or complex coda. Therefore we need to posit a markedness constraint that bans a syllable from having consonant clusters:

(19) $\texttt{OCP}_\sigma$[-syllabic]: Assign a violation to any syllable σ that has a consecutive segments with [-syllabic] feature

As demonstrated in the section above, deletion and epenthesis are usually not adopted as a solution of breaking down consonant clusters. Therefore there are several equally highly ranked faithfulness

constraints that preserves as many segments as possible in the URs:

(20) DEP: Assign a violation to any segment that is epenthesized

(21) MAX: Assign a violation to any segment that is deleted from UR

Since gemination of a consonant where a consonant bears a mora to become the nucleus of a syllable is the main strategy, a lower ranked faithfulness constraint that bans gemination is required:

(22) $*\mu-$C: Assign a violation to any consonant that bears a mora

We also need to explain why the next syllable dominates a consonant directly so that it has an onset. A constraint that bans onsetless syllables is required:

(23) ONSET: Assign a violation to any syllable without an onset

Here is the partial order of the constraints above where ONSET is not ranked:

(24) OCP_σ[-syllabic], DEP, MAX $>>$ $*\mu-$C; ONSET

The word "eagle" with UR /ʔsaː/ and SR [ʔs̪.saː] is used as an example with relevant candidates to demonstrate how the constraint interactions generate the observed SR:

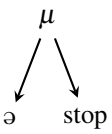
(25)

/ʔsaː/	OCP_σ[-syllabic]	DEP	MAX	$*\mu-$C	ONSET
☞[ʔs̪.saː]				*	
[ʔs̪.aː]				*	!*
[ʔs.saː]	!*				
[ʔə.saː]		!*			
[saː]			!*		

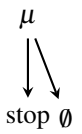
The first candidate violates the lower ranked constraint $*\mu-$C. The second candidate violates both the constraint $*\mu-$C and ONSET and thereby harmonically bounded by the first constraint. The last three candidate each violates a highly ranked constraint OCP_σ[-syllabic], DEP, and MAX and are thereby all knocked out. The winner is the observed SR [ʔs̪.saː].

Notice that when the second consonant C_2 in the consonant cluster bears the feature [-continuant], *i.e.* when C_2 is a stop, a schwa [ə] is epenthesized and forms together with C_1 and C_2 a syllable. If a stop can bear a mora alone, same structure would be posited for the stop and then the epenthesized [ə] cannot be explained. If we posit a new constraint that bans a stop from bearing a mora, while the epenthesized [ə] has to bear the mora, then we would not be able to explain why the stop is still geminated. Therefore, my proposal is that the mora is indeed shared by the [ə] and the stop as in (25), while (26) and (27) should be banned.

(26)



(27)



(28)

The sonority hierarchy can explain the left structure. Dell & Elmedlaoui (1985, 1988, 1989) (a simpler version is also proposed by Harris 1989b) discovered that assignment of nuclear status is determined by the relative sonority of the elements in the string. More sonorant segments are preferred to be syllable nucleus to less sonorant segments. Since Yavapai only has voiceless stops and voiceless stops are the least sonorant segments of all, they are the least preferred segments to be the nucleus of syllables. Therefore we propose a markedness constraint that bans a voiceless stop from being the nucleus of a syllable:

(29) *NUC/LS: Assign a violation to any voiceless stop being the nucleus of a syllable.

Sonority hierarchy can also explain the right structure. Based on the vocalic portion of sonority hierarchy (de Lacy 2006: 68), [ə] is the least sonorant vowel segment in Yavapai. We can propose

a similar constraint to ban a schwa from being the nucleus of a syllable as the constraint that bans stops from being nuclei:

(30) `*NUC/ə`: Assign a violation to any schwa that is the nucleus of a syllable.

Since schwa [ə] is allowed to be epenthesized when the consonant is a stop, the constraint `DEP` needs to be decomposed into `DEP/ə` and `DEP(ə)` in which the former punishes all epenthesized segments except [ə] and the latter punishes only the epenthesized [ə]. `DEP(ə)` is ranked lower than $\texttt{OCP}_\sigma$`[-syllabic]` because schwa is tolerated when necessary.

`*NUC/LS` is ranked higher than `DEP(ə)` because schwa is epenthesized so that it would not be violated. `*NUC/ə` is ranked higher than $*\mu-$`C` because the latter is violated so that the formed syllable conforms to `*NUC/ə`.

We use the word "it hurts" with UR /tki/ and SR [təkˀ.ki] as an example:

(31)

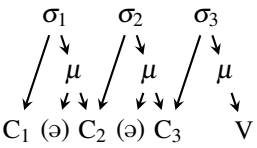
/tki/	$\texttt{OCP}_\sigma$`[-syllabic]`	`*NUC/LS`	`*NUC/ə`	$*\mu-$`C`	`DEP(ə)`
☞[təkˀ.ki]				*	*
[tḵ.ki]		!*		*	
[tə.ki]			!*		*
[tk.ki]	!*				

The first candidate is the winner because all of the rest violate constraints that are ranked higher.

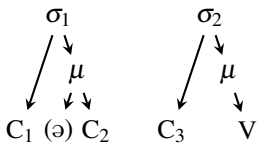
3.1 Optionality and Frequency

After explaining the basic syllabification and gemination phenomenon, we explicate in the following that OT alone is not capable of generating optionality with correct prediction of the frequencies. Below are the two moraic structures for two possible surface realizations of $C_1C_2C_3V$:

(32)



(33)



Based on these two structures, we can see that the optionality is indeed the optionality of whether C_2 and C_3 form a syllable. It is reasonable since both the structures construct well-formed syllables based on the constraints we have on syllable structure: all the syllables of both structures have onset, dominate at least one mora and do not have a single mora or a single stop being the syllable nucleus. However, based on the current set of constraints, the long form is harmonically bounded by the short ones. We use the word "band for cradle" with UR /smpuːrβi/, the short SR [sm̩.puː.βi], and the long SR [sm̩.məpˀ.puː.βi] as an example:

(34)

/smpuːrβi/	$\texttt{OCP}_\sigma$`[-syllabic]`	$*\mu-$`C`	`DEP(ə)`
☞[sm̩.puː.βi]		*	
[sm̩.məpˀ.puː.βi]		*!*	*

The long SR is harmonically bounded by the short SR because the extra syllable formed between C_2 and C_3 necessarily incurs violation on the lower ranked faithfulness constraints more. Therefore, a new constraint that favors the long SR needs to be proposed. One such option is `CodaCond`Junko (1986) which indicates codas are disallowed unless linked to a following onset:

(35) `CodaCond`: Assign a violation to two adjacent syllables if the coda and onset do not agree on place of articulation

This constraint must be ranked higher than $*\mu-$`C` that so that it favors the long SR over the short one. However, with this new constraint, the long SR would not be the optimal candidate either:

(36)

/smpuːrβi/	OCP$_\sigma$[-syllabic]	CodaCond	$*\mu-$C	DEP(ə)
☞[sm̩.məp˺.puːr.rβ̩.βi]			***	*
[sm̩.puːr.βi]		!**	*	
[sm̩.məp˺.puːr.βi]		!*	**	*

A new candidate *[sm̩.məp˺.puːr.rβ̩.βi] which has a syllable generated between /r/ and /β/ is optimal with no violation against CodaCond, while the observed long SR violates this constraint once and the short SR violates it twice. Indeed, Yavapai generally tolerates adjacent syllables with unrelated coda and onset as shown in (8).

3.2 Summary

This section adopts the traditional OT analysis to explain the syllabification process. The following constraints are proposed:

(37) OCP$_\sigma$[-syllabic], DEP/ə, MAX, *NUC/LS, *NUC/ə, $*\mu-$C, DEP(ə), ONSET

However, under current set of constraints, the long form is always harmonically bounded. No natural OT constraints or processes have been proposed to solve the dilemma incurred by the long forms being an optimal candidate. Since the correct patterns of the two forms cannot be generated by OT naturally, the frequencies of the two forms cannot be explained either. Neither the problem of optionality nor frequencies is solved. In the section below, I adopt the framework of subregular program to solve these two problems.

4 Subsequential Functions Explanation

To solve the two puzzles above, we present two functions f_{long}, f_{short} taking URs as inputs that are able to generate the long SRs and the short SRs respectively. The free variance can be explained assuming that both functions are accessible in the speakers' mind. It can also be proven that the computational complexity of f_{long} is much simpler than that of f_{short} SR, *i.e.*. f_{short} belongs to a class of functions whose computational/expressive power is stronger than the class that f_{short} belongs to. Though the notion of complexity is formal and mathematical, some experiments have shown that human cognition is sensitive to formal complexity (Avcu et al. 2019). We hypothesize that when both SRs are well-formed based on the syllable phonotactics, the one less complex to generate is more frequently attested. Given more than one possible SR for a given UR, the frequency of an SR's attestation is a function of its computational complexity.

4.1 Subregular Program

Much literature since Chomsky (1956) has been devoted to computational properties of language patterns and focuses on identifying the expressive power of the grammars that generate them. To measure their computational complexity is to classify the pattern on the Chomsky hierarchy of formal languages. The work of Johnson (1972), Kaplan and Kay (1994), Koskenniemi (1983) narrowed the computational complexity bound of phonological processes to regular and Heinz (2018, etc) proposed subregular hierarchy to furthermore classify regular functions into more fine-grained computational classes, shown in Figure 1.

There are two sub-classes of regular relations – left-subsequential and right-subsequential functions – which represent functions that read in segments of input strings from either left or right and output deterministically. Less complex than the subsequential class of functions are left Output Strictly Local (OSL) and right OSL functions, representing functions whose SR segments are determined by a finite number of previous SR segments. The intersection of left OSL and right OSL functions are Input Strictly Local (ISL) functions, representing functions whose SR segments are determined by a finite number of previous UR segments. We will prove that f_{short} is a left-subsequential function while f_{long} is an ISL function, *i.e.* f_{short} is more complex than f_{long}.

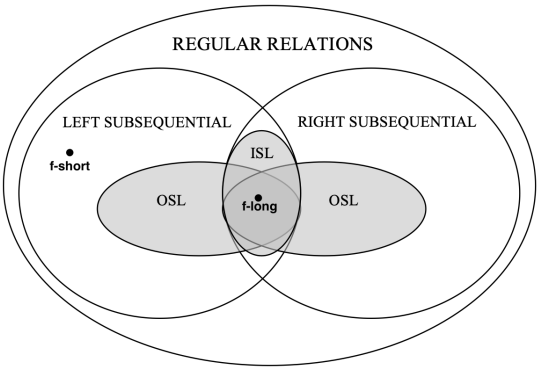


Figure 1: subregular hierarchy of functions.

4.2 Functions and their Computational Complexity

First, we present the two functions represented by a formal tool – finite state transducers (FSTs). An FST encodes input-output pairs (UR-SR pairs) with the states being limited memories of previous underlying or surface segments. It reads in the input segment in the UR string from left to right one by one and outputs the corresponding SR strings. The same input has a different output based on the current state it is in and for each input/output, a transition brings the transducer into the next state.

Definition 1 (Finite State Transducer)**.** *A finite transducer T is a 6-tuple $(Q,\Sigma,\Gamma,q_0,F,\delta)$ such that: Q is a finite set, the set of states; Σ is a finite set, called the input alphabet; Γ is a finite set, called the output alphabet; $q_0 \in Q$ and is the initial state; F is a subset of Q, the set of final states; and $\delta \subseteq Q \times (\Sigma \cup \{\rtimes, \ltimes, \lambda\}) \times (\Gamma \cup \{\lambda\}) \times Q$ (where λ is the empty string) is the transition relation.*

The following FST in Figures 2 and 3 generate the long form and short form respectively. To prove the complexities of the two FSTs, below are some useful definitions.

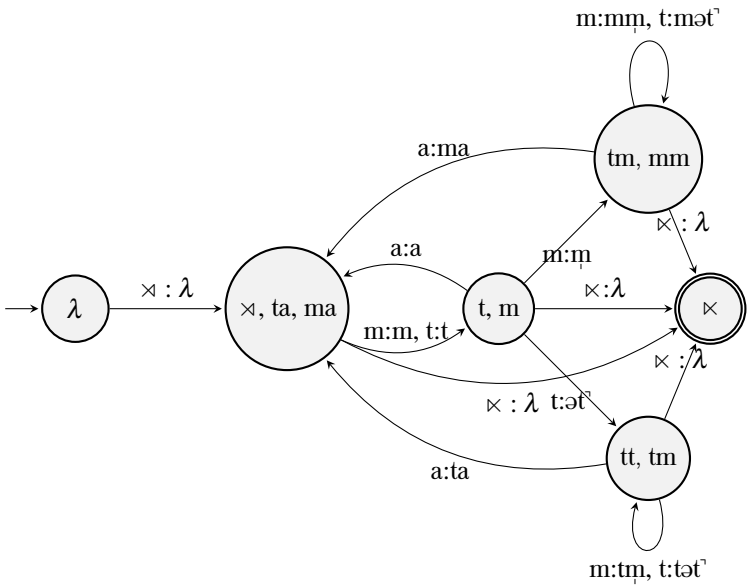


Figure 2: FST generating the long form.

For the simplicity of explication, suppose the underlying input alphabet Σ is {t, m, a}. $\rtimes$ and $\ltimes$ represent the beginning and ending of UR respectively. Using this transducer that maps /mtat/ $\rightarrow$ [məˀ.tat] /mttat/ $\rightarrow$ [məˀ.təˀ.tat], the phonological pattern from URs to long forms is generated. If a comprehensive alphabet is used in the transducer, correct long forms can be generated for all UR strings.

Definition 2 (Tails). *For $x \in \Sigma^*$ and $f : \Sigma^* \to \Sigma^*$ being a function, $\texttt{tails}_f(x) = \{(y,v) \mid f(xy) = uv \wedge u = \texttt{lcp}(f(x\Sigma^*))\}$ where $\texttt{lcp}(f(x\Sigma^*))$ calculates the longest common prefix of the set of strings $f(x\Sigma^*)$.*

The tails are the contributions to the output string of all possible extensions of $x \in \Sigma^*$ (Chandlee 2014). Take the transducer in Figure 2 as an example. Let f be the function that maps URs to their long SRs. $\texttt{tails}_f(\rtimes m) = \{(y,v) \mid f(\rtimes my) = uv \wedge u = \texttt{lcp}(f(\rtimes \text{m}\Sigma^*))\}$ where $\texttt{lcp}(f(\rtimes \text{m}\Sigma^*)) = \texttt{lcp}(\{f(\rtimes\text{ma}), f(\rtimes\text{ mt}), f(\rtimes\text{mm}), f(\rtimes\text{mat}), ...\}) = m$. Thereby $\texttt{tails}_f(\rtimes\text{m}) = \{(y,v) \mid f(\rtimes\text{my}) = uv \wedge u = \text{m}\} = \{(\rtimes, \lambda), (\text{m, m̩}), (\text{a, a}), (\text{t, ət}^\text{ˀ}), ...\}$.

Definition 3 (Input Strictly Local Function). *A function f is Input Strictly Local (ISL) iff $\exists k \in \mathcal{N}$ s.t. for all $u_1, u_2 \in \Sigma^2$, it is the case that if $\texttt{suff}_{k-1}(u_1) = \texttt{suff}_{k-1}(u2)$ then $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$.*

Using the definitions above, it can be proven that the FST that generates the long form pattern must encode f an ISL_3 function. The lemma below (Chandlee 2014:40, Theorem 2) is helpful for the proof.

Lemma 1. *A function f is ISL iff $\exists k \in \mathcal{N}$ s.t. f can be described with an FST for which $Q = \Sigma^{k-1}$ and $\forall q_1, q_2 \in Q, \sigma_1, \sigma_2 \in \Sigma, u_1, u_2 \in \Gamma$, if $(q_1, \sigma_1, u_1, q_1'), (q_2, \sigma_2, u_2, q_2') \in \delta$ and $\texttt{suff}_{k-1}(q_1\sigma_1) = \texttt{suff}_{k-1}(q_2\sigma_2)$, then $q_1' = q_2'$.*

The first part of the lemma requires that the states of the FSTs that describe ISL_k functions correspond to the set of possible string of length $k-1$. The second part indicates that if two input strings $q_1\sigma_1, q_2\sigma_2$ agree on the $k-1$ segments, then they end on the same state $q_1', q_2' \in Q, q_1' = q_2'$.

Theorem 1. *The transformation between URs and their long SRs is an ISL_3 function.*

Proof. To prove that the transformation is an ISL_3 function, we need to prove that it is not ISL_2 and an ISL_3 transducer represents the transformation. The transformation is not ISL_2 based on Definition 3: Let $u_1 =$/m/, $u_2 =$/mm/. While $\texttt{suff}_{2-1}(u_1) = \texttt{suff}_{2-1}(u_2) =$m, $\texttt{tails}_f(\text{m}) = \{(\ltimes, \lambda), (\text{m, m̩}), (\text{t, ət}^\text{ˀ}), (\text{a, a}), ...\} \neq \texttt{tails}_f(\text{mm}) = \{(\ltimes, \lambda), (\text{m, mm̩}), (\text{t, mət}^\text{ˀ}), (\text{a, ma}), ...\}$. Therefore the transformation is not ISL_2. To show that it is ISL_3, we can show that the function represented by the FST in Figure 2 is ISL_3. Based on Lemma 1, all $q \in Q$ are elements in Σ^2 and for all $u_1, u_2 \in \Sigma^*$, if $\texttt{suff}_2(u_1) = \texttt{suff}_2(u_2)$, then they both reach the same state. For example, input strings /mtta/ and /mtata/ both reach the state "$\rtimes$, ta, ma". Thereby the transformation between URs and long SRs is an ISL_3 function. □

Definition 4 (Output Strictly Local Function). *A function f is Output Strictly Local iff $\exists k \in \mathcal{N}$ s.t. for all $u_1, u_2 \in \Sigma^*$, it is the case that if $\texttt{suff}_{k1}(f(u_1)) = \texttt{suff}_{k1}(f(u_2))$ then $\texttt{tails}_f(u_1) = \textit{tails}_f(u_2)$.*

Now we present the FST Figure 3 that generates the short form. This figure only depicts half of the transducer for clear demonstration. The other half is symmetric to this presented part. Using this transducer that maps /tmta/→[tm̩.ta], /tmmmatma/→[tm̩.mm̩.mat.ma] and *etc.*, the expected short forms are correctly generated. However, the function that maps UR strings to their corresponding short SRs is not an ISL function.

Theorem 2. *The transformation f between URs and their short SRs is not an ISL function.*

Proof. Proof by contradiction. Suppose the transformation is an ISL_k function for some even integer k. Let $u_1 =$/mm...mt/ with $k-2$ m and one /t/ in the end, $u_2 =$/tamm...mt/ with ta in the beginning, $k-2$ m in between and one t in the end. $\texttt{suff}_{k-1}(u_1) = \texttt{suff}_{k-1}(u_2) =$/mm...mt/. Then $\texttt{lcp}(f(\text{m...mt}\Sigma^*)) = [\text{mm̩...mm̩}]$ with $\frac{k-2}{2}$ many mm̩ in the string because $f(\text{m...mta}) = [\text{mm̩...mm̩.ta}]$ while $f(\text{m...mt}) = [\text{mm̩...mm̩.mət}^\text{ˀ}]$. $\texttt{lcp}(f(\text{tam...mt}\Sigma^*)) = [\text{tam.mm̩...mm̩.mət}^\text{ˀ}]$ with $\frac{k-4}{2}$ many [mm̩] in the string. Then $\texttt{tails}_f(\text{m...mt}) = \{(\ltimes, \text{mət}^\text{ˀ}), (\text{t, tət}^\text{ˀ}), (\text{ta, tət}^\text{ˀ}\text{.ta}), ...\} \neq \texttt{tails}_f(\text{tam...mt}) = \{(\ltimes, \lambda), (\text{t, tət}^\text{ˀ}), (\text{ta, ta}), ...\}$. Thereby the transformation f is not an ISL_k function for any even integer k. Suppose the transformation is an ISL_k function for some odd integer k. Then $\texttt{lcp}(f(\text{m...mt}\Sigma^*)) = [\text{mm̩...mm̩.mət}^\text{ˀ}]$ with $\frac{k-3}{2}$ many mm̩ in the string, $\texttt{lcp}(f(\text{tam...mt}\Sigma^*)) = [\text{tam.mm̩...mm̩}]$ with $\frac{k-3}{2}$

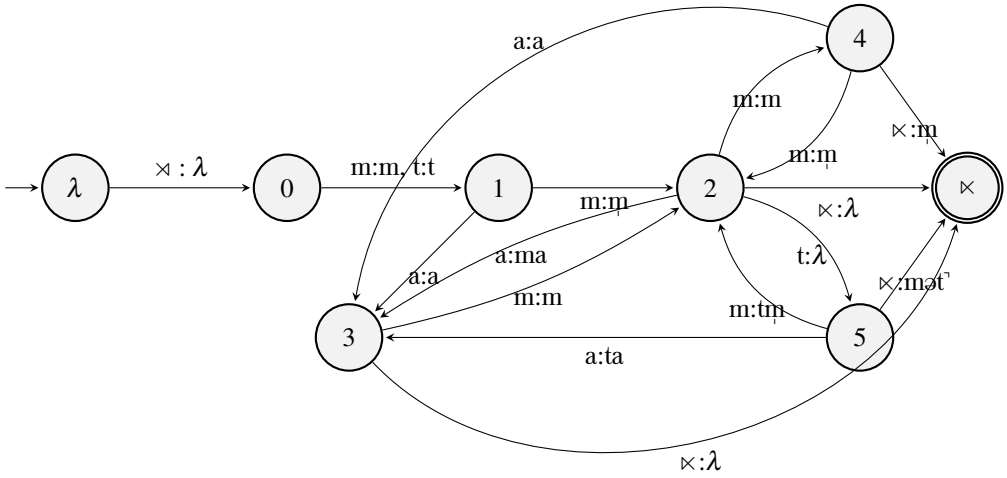


Figure 3: FST generating the short form.

many [mm̥] in the string. Then $\texttt{tails}_f$(m...mt) = {(⋉, λ), (t, təʔ), (ta, ta), ...} ≠ $\texttt{tails}_f$(tam...mt) = {(⋉, məʔ), (t, təʔ), (ta, təʔ.ta), ...}. Thereby the transformation f is not an ISL$_k$ function for any odd integer k. Therefore the transformation f is not an ISL$_k$ function for any integer k and it is not an ISL function. □

Indeed, we can show that this transformation function is not an OSL function as well, which result combined with Theorem 3 implicates that f_{short} is more complex than f_{long}.

Theorem 3. *The transformation f between URs and their short SRs is not an OSL function.*

Proof. Proof by contradiction. Suppose the transformation is an OSL$_k$ function for some even integer k. Let u_1 =/tamm...m/ for $k+1$ segments m and u_2 =/tamm...m/ for k segments m. Both strings have surface representation *[tam.mm̥....mm̥]* with $\frac{k}{2}$ many [mm̥] and $\texttt{suff}_{k-1}(f(u_1))$ = $\texttt{suff}_{k-1}(f(u_2))$ = m̥...mm̥. However, $\texttt{tails}_f(u_1) \neq \texttt{tails}_f(u_2)$ because $\texttt{lcp}(f(u_1\Sigma^*))$ = [tam.mm̥... mm̥] with $\frac{k}{2}$ many [mm̥] so that $\texttt{tails}_f(u_1)$ = {(⋉, λ), (t), məʔ), ...}, while $\texttt{lcp}(f(u_1\Sigma^*))$ = [tam.mm̥...mm̥] with $\frac{k-2}{2}$ many [mm̥] so that $\texttt{tails}_f(u_2)$ = {(⋉, mm̥), (t), məʔ), ...}. Thereby the transformation is not an OSL$_k$ function for any even integer k. Now suppose the transformation is an OSL$_k$ function for some odd integer k. Again, let u_1 =/tamm...m/ for $k+1$ segments m and u_2 =/tamm...m/ for k segments m. A very similar procedure as above can be used to prove that even though $\texttt{suff}_{k-1}(f(u_1))$ = $\texttt{suff}_{k-1}(f(u_2))$ =mm̥...mm̥, $\texttt{tails}_f(u_1) \neq \texttt{tails}_f(u_2)$. Thereby f is not an OSL$_k$ function for any odd integer k and therefore the transformation is not an OSL function. □

Since the phonological transformation f_{shot} from URs to short SRs is neither an ISL function nor an OSL function, it is computationally more complex than the phonological transformation f_{long} from URs to long SRs which is an ISL$_3$ function. Indeed, it can be proven that f_{long} is an OSL$_2$ function as well, thereby $f_{long} \in ISL \cap OSL$ while $f_{short} \notin ISL \cup OSL$. It can be easily seen that f_{short} is a left-subsequential function which is defined as the set of functions that can be described by a deterministic finite state transducer reading inputs from left to right. Since FST in Figure 3 is deterministic, *i.e.* the transition of each input segment gives exactly one output and brings the transducer to one specific state. Since f_{long} is both ISL and OSL while f_{short} is left-subsequantial but neither ISL nor OSL, f_{short} is computationally more complex than f_{long}.

The two functions that generate the two surface representations solve the problem of (1) correctly generating the two forms (2) explaining the relative frequencies of the attested forms. The

two finite state transducers in Figure 2 and Figure 3 correctly generate the two forms. In addition, based on our hypothesis that when both SRs are well-formed based on the syllable phonotactics, the one less complex to generate is more frequently attested. Therefore the computational complexity hierarchy encoded inherently in the two functions explains the different frequencies attested.

5 Conclusion

In this paper, we present the syllabification process among consonant clusters in Yavapai, which aims at avoiding adjacent consonants to occur in one syllable in the surface representations. While substance-based OT framework demonstrates how one syllable is formed and the phonotactics of syllables, it cannot generate the free variation between the long form and the short form. It thereby cannot explain the different frequencies attested either. The computational approach complements the OT approach in that it provides subsequential functions that generate both the two surface representations and explains away the frequency problem by the computational complexity hierarchy between them. The syllabification instance in Yavapai illustrates that computation also matters in phonology in addition to substance.

References

Avcu, Enes, Ryan Rhodes, and Arild Hestvik. 2019. Neural underpinnings of phonotactic rule learning. In *Proceedings of the Annual Meetings on Phonology*, volume 7.

Chandlee, Jane. 2014. Strictly Local Phonological Processes. Doctoral dissertation, University of Delaware.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2:113–124.

Davis, Stuart. 2011. Geminates. *The Blackwell Companion to Phonology* 1–25.

Hayes, Bruce. 1989. Compensatory lengthening in moraic phonology. *Linguistic Inquiry* 20:253–306.

Heinz, Jeffrey. 2018. The computational nature of phonological generalizations. In *Phonological Typology*, ed. L. Hyman and F. Plank, Phonetics and Phonology, chapter 5, 126–195. De Gruyter Mouton.

Jardine, Adam. 2019. Computation also matters: a response to Pater (2018). *Phonology* 36:341–350.

Johnson, C. Douglas. 1972. *Formal Aspects of Phonological Description*. The Hague: Mouton.

Junko, Itô. 1986. *Syllable Theory in Prosodic Phonology*. University Microfilms International [Publisher].

Kaplan, Ronald M, and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20:331–378.

Koskenniemi, Kimmo. 1983. Two-level model for morphological analysis. In *IJCAI*, volume 83, 683–685.

Ridouane, Rachid. 2010. Geminates at the junction of phonetics and phonology. *Laboratory phonology* 10:61–90.

Shaterian, Alan. 1983. Phonology and Dictionary of Yavapai. Doctoral dissertation, University of California, Berkeley.

Tranel, Bernard. 1991. CVC light syllables, geminates and moraic theory. *Phonology* 8:291–302.

Department of Linguistics
Rutgers Universiy, New Brunswick
New Brunswick, New Jersey, 08901
wenyue.hua@rutgers.edu