Finite-state Model of Shupamem Reduplication

Magdalena Markowska, Jeffrey Heinz, and Owen Rambow
Stony Brook University
Department of Linguistics &
Institute for Advanced Computational Science
{magdalena.markowska,jeffrey.heinz,owen.rambow}@stonybrook.edu

Abstract

Shupamem, a language of Western Cameroon, is a tonal language which also exhibits the morpho-phonological process of full reduplication. This creates two challenges for a finite-state model of its morpho-syntax and morpho-phonology: how to manage the full reduplication, as well as the autosegmental nature of lexical tone. Dolatian and Heinz (2020) explain how 2-way finite-state transducers can model full reduplication without an exponential increase in states, and finite-state transducers with multiple tapes have been used to model autosegmental tiers, including tone (Wiebe, 1992; Dolatian and Rawski, 2020a; Rawski and Dolatian, 2020). Here we synthesize 2-way finite-state transducers and multi-tape transducers, resulting in a finite-state formalism that subsumes both, to account for the full reduplicative processes in Shupamem which also affect tone.

1 Introduction

Reduplication is a very common morphological process cross-linguistically. Approximately 75% of world languages exhibit partial or total reduplication (Rubino, 2013). This morphological process is particularly interesting from the computational point of view because it introduces challenges for 1-way finite-state transducers (FSTs). Even though partial reduplication can be modelled with 1-way FSTs (Roark and Sproat, 2007; Chandlee and Heinz, 2012), there is typically an explosion in the number of states. Total reduplication, on the other hand, is the only known morpho-phonological process that cannot be modelled with 1-way FSTs because the number of copied elements, in principle, has no upper bound. Dolatian and Heinz (2020) address this challenge with 2-way FSTs, which can move back and forth on the input tape, producing a faithful copy of a string. Deterministic 2-way FSTs can model both partial and full segmental reduplication in a compact way.

However, many languages that exhibit reduplicative processes also are tonal, which often means that tones and segments act independently from one another in their morpho-phonology. For instance, in Shupamem, a tonal language of Western Cameroon, *ndáp* 'house' → *ndâp ndàp* 'houses' (Markowska, 2020).

tones	H	HL L
segments	ndap	ndap ndap

Pioneering work in autosegmental phonology (Leben, 1973; Williams, 1976; Goldsmith, 1976) shows tones may act independently from their tone-bearing units (TBUs). Moreover, tones may exhibit behavior that is not typical for segments (Hyman, 2014; Jardine, 2016), which brings yet another strong argument for separating them from segments in their linguistic representations. Such autosegmental representations can be mimicked using finite-state machines, in particular, Multi-Tape Finite-State Transducers (MT FSTs) (Wiebe, 1992; Dolatian and Rawski, 2020a; Rawski and Dolatian, 2020). We note that McCarthy (1981) uses the same autosegmental representations in the linguistic representation to model templatic morphology, and this approach has been modeled for Semitic morphological processing using multi-tape automata (Kiraz, 2000; Habash and Rambow, 2006).

This paper investigates what finite-state machinery is needed for languages which have both reduplication and tones. We first argue that we need a synthesis of the aforementioned transducers, i.e. 1-way, 2-way and MT FSTs, to model morphology in the general case. The necessity for such a formal device will be supported by the morpho-phonological processes present in Shupamem nominal and verbal reduplication. We then discuss an

Proceedings of the Seventeenth SIGMORPHON Workshop on Computational Research

in Phonetics, Phonology, and Morphology,pages 212–221

alternative, in which we use the MT FST to handle both reduplication and tones. It is important to emphasize that all of the machines we discuss are deterministic, which serves as another piece of evidence that even such complex processes like full reduplication can be modelled with deterministic finite-state technology (Chandlee and Heinz, 2012; Heinz, 2018).

This paper is structured as follows. First, we will briefly summarize the linguistic phenomena observed in Shupamem reduplication (Section 2). We then provide a formal description of the 2-way (Section 3) and MT FSTs (Section 4). We propose a synthesis of the 1-way, 2-way and MT FSTs in Section 5 and further illustrate them using relevant examples from Shupamem in Section 6. In Section 7 we discuss a possible alternative to the model which uses only MT FSTs. Finally, in Section 8 we show that the proposed model works for other tonal languages as well, and we conclude our contributions.

2 Shupamem nominal and verbal reduplication

Shupamem is an understudied Grassfields Bantu language of Cameroon spoken by approximately 420,000 speakers (Eberhard et al., 2021). It exhibits four contrastive surface tones (Nchare, 2012): high (H; we use diacritic V́ on a vowel as an orthographic representation), low (L; diacritic V̀), rising (LH; diacritic V̌), and falling (HL; diacritic V̂). Nouns and verbs in the language reduplicate to create plurals and introduce semantic contrast, respectively. Out of 13 nouns classes, only one exhibits reduplication. Nouns that belong to that class are monosyllabic and carry either H or L lexical tones. Shupamem verbs are underlyingly H or rising (LH). Table 1 summarizes the data adapted from Markowska (2020).

In both nouns and verbs, the first item of the reduplicated phrase is the base, while the reduplicant is the suffix. We follow the analysis in Markowska (2020) and summarize it here. The nominal reduplicant is toneless underlyingly, while the verbal reduplicant has an H tone. Furthermore, the rule of Opposite Tone Insertion explains the tonal alternation in the base of reduplicated nouns, and Default L-Insertion accounts for the L tone on the suffix. Interestingly, more tonal alternations are observed when the tones present in the reduplicated phrase interact with other phrasal/grammatical tones. For the purpose of this paper, we provide only a summary of those tonal alternations in Table 2.

	Transl.	Lemma	Red form
Nouns		H	HL L
	‘crab’	*kám*	*kâm kàm*
		L	LH L
	‘game’	*kàm*	*kăm kàm*
Verbs		H	H$^{\downarrow}$H
	‘fry’	*ká*	*ká k*$^{\downarrow}$*á*
		LH	LH$^{\downarrow}$H
	‘peel’	*kă*	*kă k*$^{\downarrow}$*á*

Table 1: Nominal and verbal reduplication in Shupamem

	Red. tones	Output
Nouns	HL L	HL <u>H</u>
	LH L	LH <u>H</u>
Verbs	H$^{\downarrow}$H	H$^{\downarrow}$H
	LH$^{\downarrow}$H	<u>HL LH</u>

Table 2: Tonal alternations: interaction of tones for reduplicated forms (“Red. tones”) with grammatical H tones

The underlined tones indicate changes triggered by the H grammatical/phrasal tone. In the observance of H tone associated with the right edge of the subject position in Shupamem, the L tone that is present on the surface in the suffix of reduplicated nouns (recall Table 1), now is represented with an H tone. Now it should be clear that the noun reduplicant should, in fact, be toneless in the underlying representation (UR). While the presence of H tone directly preceding the reduplicated verb does not affect H-tone verbs, such as *ká* ‘fry’, it causes major tonal alternations in rising reduplicated verbs. Let us look at a particular example representing the final row in Table 2:

pɔ́ ‘PASTIII’ + *kă ká* ‘peel.CONTR’ → *pɔ́ kâ kă*

The H tone associated with the tense marker introduces two tonal changes to the reduplicated verb: it causes tonal reversal on the morphological base, and it triggers L-tone insertion to the left edge of the reduplicant.

The data in both Table 1 and 2 show that 1) verbs and nouns in Shupamem reduplicate fully at the segmental level, and 2) tones are affected by phonological rules that function solely at the suprasegmental level. Consequently, a finite-state model of the language must be able to account for

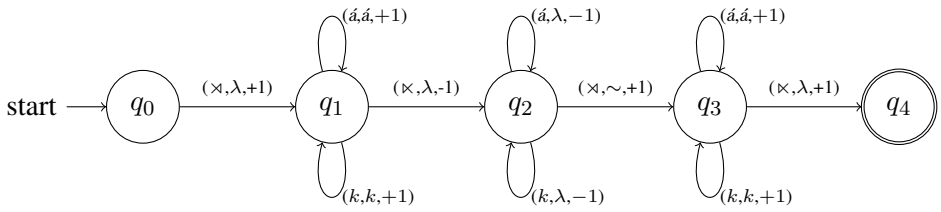

Figure 1: 2-way FST for total reduplication of *ká* 'fry.IMP → *ká ká* 'fry.IMP as opposed to boiling'

those factors. In the next two sections, we will provide a brief formal introduction to 2-way FSTs and MT-FSTs, and explain how they correctly model full reduplication and autosegmental representation, respectively.

In this paper, we use an orthographic representation for Shupamem which uses diacritics to indicate tone. Shupamem does not actually use this orthography; however, we are interested in modeling the entire morpho-phonology of the language, independently of choices made for the orthography. Furthermore, many languages do use diacritics to indicate tone, including the Volta-Niger languages Yoruba and Igbo, as well as Dschang, a Grassfields language closely related to Shupamem. (For a discussion of the orthography of Cameroonian languages, with a special consideration of tone, see (Bird, 2001).) Diacritics are also used to write tone in non-African languages, such as Vietnamese. Therefore, this paper is also relevant to NLP applications for morphological analysis and generation in languages whose orthography marks tones with diacritics: the automata we propose could be used to directly model morpho-phonological computational problems for the orthography of such languages.

3 2-way FSTs

As Roark and Sproat (2007) point out, almost all morpho-pholnological processes can be modelled with 1-way FSTs with the exception of full reduplication, whose output is not a regular language. One way to increase the expressivity of 1-way FST is to allow the read head of the machine to move back and forth on the input tape. This is exactly what 2-way FST does (Rabin and Scott, 1959), and Dolatian and Heinz (2020) explain how these transducers model full reduplication not only effectively, but more faithfully to linguistic generalizations.

Similarly to a 1-way FST, when a 2-way FST reads an input, it writes something on the output tape. If the desired output is a fully reduplicated string, then the FST faithfully 'copies' the input string while scanning it from left to right. In contrast, while scanning the string from right to left, it outputs nothing (λ), and it then copies the string again from left to right.

Figure 1 illustrates a deterministic 2-way FST that reduplicates *ká* 'fry.IMP'; readers are referred to Dolatian and Heinz (2020) for formal definitions. The key difference between deterministic 1-way FSTs and deterministic 2-way FSTs are the addition of the 'direction' parameters $\{+1, 0, -1\}$ on the transitions which tell the FST to advance to the next symbol on the input tape (+1), stay on the same symbol (0), or return to the previous symbol (-1). Deterministic 1-way FSTs can be thought of as deterministic 2-way FSTs where transitions are all (+1).

The input to this machine is $\rtimes$*ká*$\ltimes$. The $\rtimes$ and $\ltimes$ symbols mark beginning and end of a string, and $\sim$ indicates the boundary between the first and the second copy. None of those symbols are essential for the model, nevertheless they facilitate the transitions. For example, when the machine reads $\ltimes$, it transitions from state q_1 to q_2 and reverses the direction of the read head. After outputting the first copy (state q_1) and rewinding (state q_2), the machine changes to state q_3 when it scans the left boundary symbol $\rtimes$ and outputs $\sim$ to indicate that another copy will be created. In this particular example, not marking morpheme boundary would not affect the outcome. However, in Section 5, where we propose the fully-fledged model, it will be crucial to somehow separate the first from second copy.

4 Multitape FSTs

Multiple-tape FSTs are machines which operate in the exact same was as 1-way FST, with one key difference: they can read the input and/or write the output on multiple tapes. Such a transducer can operate in an either synchronous or asynchronous manner, such that the input will be read on all tapes

and an output produced simultaneously, or the machine will operate on the input tapes one by one. MT-FST can take a single ('linear') string as an input and output multiple strings on multiple tapes or it can do the reverse (Rabin and Scott, 1959; Fischer, 1965).

To illustrate this idea, let us look at Shupamem noun *màpàm* 'coat'. It has been argued that Shupamem nouns with only L surface tones will have the L tone present in the UR (Markowska, 2019, 2020). Moreover, in order to avoid violating the Obligatory Contour Principle (OCP) (Leben, 1973), which prohibits two identical consecutive elements (tones) in the UR of a morpheme, we will assume that only one L tone is present in the input. Consequently, the derivation will look as shown in Table 3.

Input:	T-tape	L
Input:	S-tape	mapam
Output:	Single tape	màpàm

Table 3: Representation of MT-FST for *màpàm* 'coat'

Separating tones from segments in this manner, i.e. by representing tones on the T(one)-tape and segments on the S(segmental)-tape, faithfully resembles linguistic understanding of the UR of a word. The surface form *màpàm* has only one L tone present in the UR, which then spreads to all TBUs, which happen to be vowels in Shupamem, if no other tone is present.

An example of a multi-tape machine is presented in Figure 2. For better readability, we introduce a generalized symbols for vowels (V) and consonants (C), so that the input alphabet is $\Sigma_{\rtimes} = \{(C, V), (L, H)\} \cup \{\rtimes, \ltimes\}$, and the output alphabet is $\Gamma = \{C, \acute{V}, \grave{V}\}$. The machine operates on 2 input tapes and writes the output on a single tape. Therefore, we could think of such machine as a *linearizer*. The two input tapes represent the Tonal and Segmental tiers and so we label them T and S, respectively. We illustrate the functioning of the machine using the example (mapam, L) $\rightarrow$ *màpàm* 'coat'. While transitioning from state q_0 to q_1, the output is an empty string since the left edge marker is being read on both tapes simultaneously. In state q_1, when a consonant is being read, the machine outputs the exact same consonant on the output tape. However, when the machine reaches a vowel, it outputs a vowel with a tone that is being read at the same time on the T-tape (in our example, (L, V) $\rightarrow$ V̀) and transitions to state q_2 (if the symbol on the T-tape is H) or q_3 (for L, as in our example). In states q_2 and q_3, consonants are simply output as in state q_1, but for vowels, one of three conditions may occur: the read head on the Tonal tape may be H or L, in which case the automaton transitions (if not already there) to q_2 (for H) or q_3 (for L), and outputs the appropriate orthographic symbol. But if on the Tonal tape the read head is on the right boundary marker $\ltimes$, we are in a case where there are more vowels in the Segmental tape than tones in the Tonal tape. This is when the OCP determines the interpretation: all vowels get the tone of the last symbol on the Tone tier (which we remember as states q_2 and q_3). In our example, this is an L. Finally, when the Segmental tape also reaches the right boundary marker $\ltimes$, the machine transitions to the final state q_4. This ('linearizing') MT-FST consists of 4 states and shows how OCP effects can be handled with asynchronous multitape FSTs. Note that when there are more tones on the Tonal tier than vowels on the Segmental tier, they are simply ignored. We refer readers to Dolatian and Rawski (2020b) for formal definitions of these MT transducers.

We are also interested in the *inverse* process – that is, a finite-state machine that in the example above would take a single input string [màpàm] and produce two output strings [L] and [mapam]. While multitape FSTs are generally conceived as relations over n-ary relations over strings, Dolatian and Rawski (2020b) define their machines deterministically with n input tapes and a single output tape. We generalize their definition below.

Similarly to spreading processes described above, separating tones from segments give us a lot of benefits while accounting for tonal alternations taking place in nominal and verbal reduplication in Shupamem. First of all, functions such as Opposite Tone Insertion (OTI) will apply solely at the tonal level, while segments can be undergoing other operations at the same time (recall that MT-FSTs can operate on some or all tapes simultaneously). Secondly, representing tones separately from segments make tonal processes local, and therefore all the alternations can be expresses with less powerful functions (Chandlee, 2017).

Now that we presented the advantages of MT-FSTs, and the need for utilizing 2-way FSTs to model full reduplication, we combine those machines to account for all morphophonological pro-

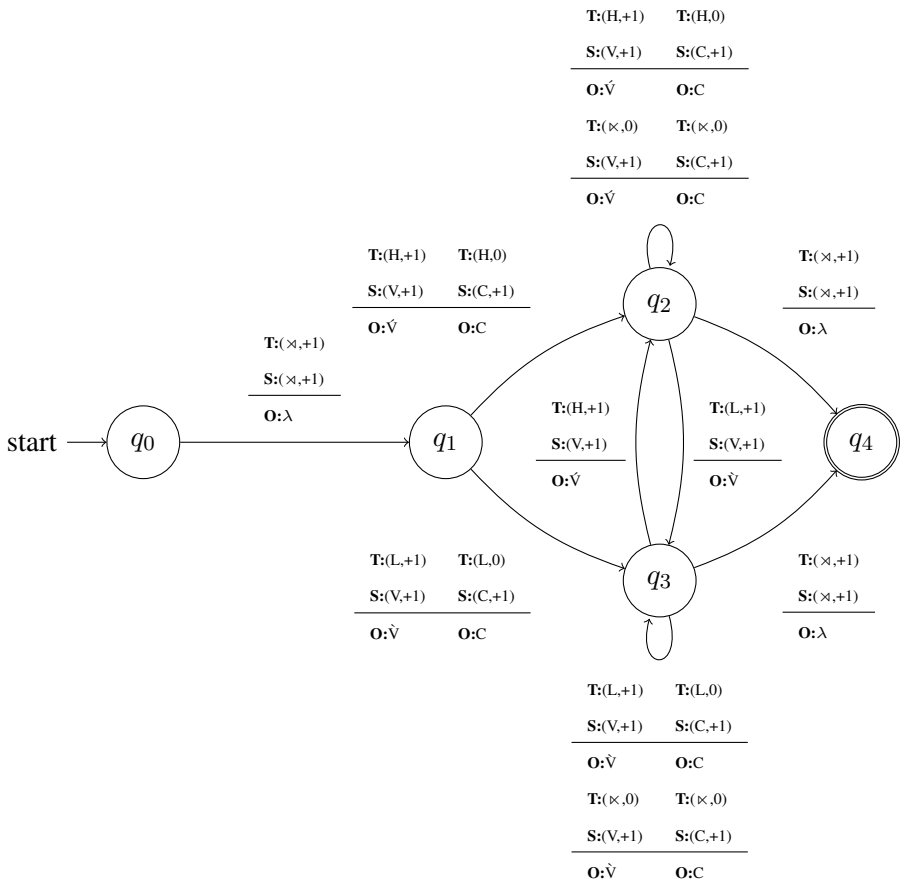

Figure 2: MT-FST: linearize

C and V are notational meta-symbols for consonants and vowels, resp.; T indicates the tone tape, S, the segmental tape, and O the output tape.

cesses described in Section 2.

5 Deterministic 2-Way Multi-tape FST

Before we define Deterministic 2-Way Multi-tape FST (or 2-way MT FST for short) we introduce some notation. An alphabet Σ is a finite set of symbols and Σ^* denotes the set of all strings of finite length whose elements belong to Σ. We use λ to denote the empty string. For each $n \in \mathbb{N}$, an n-string is a tuple $\langle w_1, \dots w_n \rangle$ where each w_i is a string belonging to $\Sigma_i^* (1 \leq i \leq n)$. These n alphabets may contain distinct symbols or not. We write $\vec{w}$ to indicate a n-string and $\vec{\lambda}$ to indicate the n-string where each $w_i = \lambda$. We also write $\vec{\Sigma}$ to denote a tuple of n alphabets: $\vec{\Sigma} = \Sigma_1 \times \cdots \Sigma_n$. Elements of $\vec{\Sigma}$ are denoted $\vec{\sigma}$.

If $\vec{w}$ and $\vec{v}$ belong to $\vec{\Sigma}^*$ then the pointwise concatenation of $\vec{w}$ and $\vec{v}$ is denoted $\vec{w}\vec{v}$ and equals $\langle w_1, \dots w_n \rangle \langle v_1, \dots v_n \rangle = \langle w_1 v_1, \dots w_n v_n \rangle$. We are interested in functions that map n-strings to m-strings with $n, m \in \mathbb{N}$. In what follows we generally use the index i to range from 1 to n and the index j to range from 1 to m.

We define Deterministic 2-Way n, m Multitape FST (2-way (n, m) MT FST for short) for $n, m \in \mathbb{N}$ by synthesizing the definitions of Dolatian and Heinz (2020) and Dolatian and Rawski (2020b); n, m refer to the number of input tapes and output tapes, respectively. A Deterministic 2-Way n, m Multitape FST is a six-tuple $(Q, \vec{\Sigma}, \vec{\Gamma}, q_0, F, \delta)$, where

- $\vec{\Sigma} = \langle \Sigma_1 \dots \Sigma_n \rangle$ is a tuple of n input alphabets that include the boundary symbols, i.e., $\{\rtimes, \ltimes\} \subset \Sigma_i,\ 1 \leq i \leq n$,
- $\vec{\Gamma}$ is a tuple of m output alphabets $\Gamma_j\ (1 \leq j \leq m)$,
- $\delta : Q \times \vec{\Sigma} \rightarrow Q \times \vec{\Gamma}^* \times \vec{D}$ is the transition function. D is an alphabet of directions equal to $\{-1, 0, +1\}$ and $\vec{D}$ is an n-tuple. $\vec{\Gamma}^*$ is a m-tuple of strings written to each output tape.

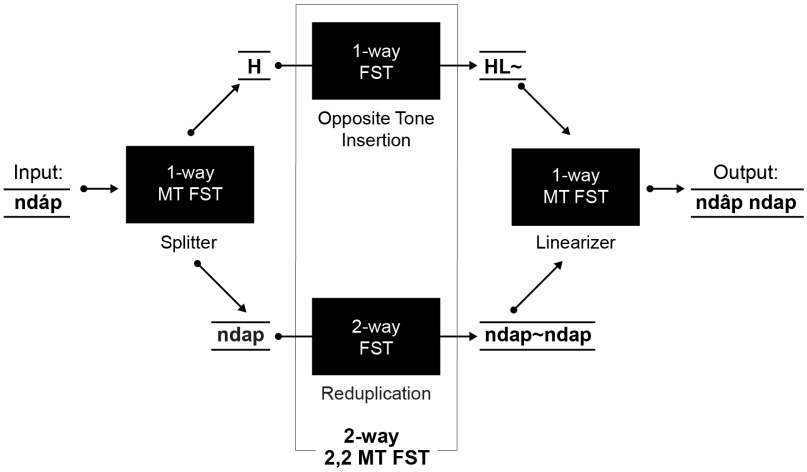

Figure 3: Synthesis of 2-way FST and MT-FST

We understand $\delta(q, \vec{\sigma}) = (r, \vec{v}, \vec{d})$ as follows. It means that if the transducer is in state q and the n read heads on the input tapes are on symbols $\langle \sigma_1, \ldots \sigma_n \rangle = \vec{\sigma}$, then several actions ensue. The transducer changes to state r and pointwise concatenates $\vec{v}$ to the m output tapes. The n read heads then move according to the instructions $\vec{d} \in \vec{D}$. For each read head on input tape i, it moves back one symbol iff $d_i = -1$, stays where it is iff $d_i = 0$, and advances one symbol iff $d_i = +1$. (If the read head on an input tape "falls off" the beginning or end of the string, the computation halts.)

The function recognized by a 2-way (n, m) MT FST is defined as follows. A *configuration* of a n, m-MT-FST T is a 4-tuple $\langle \vec{\Sigma}^*, q, \vec{\Sigma}^*, \vec{\Gamma}^* \rangle$. The meaning of the configuration $(\vec{w}, q, \vec{x}, u)$ is that the input to T is $\vec{w}\,\vec{x}$ and the machine is currently in state q with the n read heads on the first symbol of each x_i (or has fallen off the right edge of the i-th input tape if $x_i = \lambda$) and that $\vec{u}$ is currently written on the m output tapes.

If the current configuration is $(\vec{w}, q, \vec{x}, \vec{u})$ and $\delta(q, \vec{\sigma}) = (r, \vec{v}, \vec{d})$ then the next configuration is $(\vec{w}', r, \vec{x}', \vec{u}\,\vec{v})$, where for each i, $1 \leq i \leq n$:

- $\vec{w}' = \langle w_1' \ldots w_n' \rangle$ and $\vec{x}' = \langle x_1' \ldots x_n' \rangle$ $(1 \leq i \leq n)$;
- $w_i' = w_i$ and $x_i' = x_i$ iff $d_i = 0$;
- $w_i' = w_i \sigma$ and $x_i' = x_i''$ iff $d_i = +1$ and there exists $\sigma \in \Sigma_i, x_i'' \in \Sigma_i^*$ such that $x_i = \sigma x_i''$;
- $w_i' = w_i''$ and $x_i' = \sigma x_i$ iff $d_i = -1$ and there exists $\sigma \in \Sigma_i, w_i'' \in \Sigma_i^*$ such that $w_i = \sigma w_i''$.

We write $(\vec{w}, q, \vec{x}, \vec{u}) \rightarrow (\vec{w}', r, \vec{x}', \vec{u}\,\vec{v})$. Observe that since δ is a function, there is at most one next configuration (i.e., the system is deterministic). Note there are some circumstances where there is no next configuration. For instance if $d_i = +1$ and $x_i = \lambda$ then there is no place for the read head to advance. In such cases, the computation halts.

The transitive closure of $\rightarrow$ is denoted with $\rightarrow^+$. Thus, if $c \rightarrow^+ c'$ then there exists a finite sequence of configurations $c_1, c_2 \ldots c_n$ with $n > 1$ such that $c = c_1 \rightarrow c_2 \rightarrow \ldots \rightarrow c_n = c'$.

At last we define the function that a 2-way (n, m) MT FST T computes. The input strings are augmented with word boundaries on each tape. Let $\overrightarrow{\rtimes w \ltimes} = \langle \rtimes w_1 \ltimes, \ldots \rtimes w_n \ltimes \rangle$. For each n-string $\vec{w} \in \vec{\Sigma}^*$, $f_T(\vec{w}) = \vec{u} \in \vec{\Gamma}^*$ provided there exists $q_f \in F$ such that $(\vec{\lambda}, q_0, \overrightarrow{\rtimes w \ltimes}, \vec{\lambda}) \rightarrow^+ (\overrightarrow{\rtimes w \ltimes}, q_f, \vec{\lambda}, \vec{u})$.

If $f_T(\vec{w}) = \vec{u}$ then $\vec{u}$ is unique because the sequence of configurations is determined deterministically. If the computation of a 2-way MT-FST T halts on some input $\vec{w}$ (perhaps because a subsequent configuration does not exist), then we say T is undefined on $\vec{w}$.

The 2-way FSTs studied by Dolatian and Heinz (2020) are 2-way 1,1 MT FST. The n-MT-FSTs studied by Dolatian and Rawski (2020b) are 2-way n,1 MT FST where none of the transitions contain the -1 direction. In this way, the definition presented here properly subsumes both.

Figure 4 shows an example of a 1,2 MT FST that "splits" a phonetic (or orthographic) transcription of a Shupamem word into a linguistic representation

with a tonal and segmental tier by outputting two output strings, one for each tier.

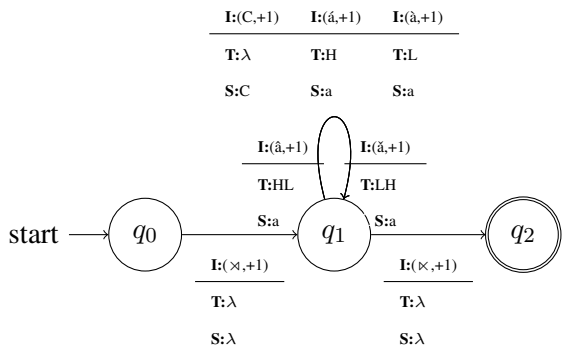

Figure 4: MT-FST: split ndáp → (ndap, H) 'house', C and V are notational meta-symbols for consonants and vowels, resp.; T indicates the output tone tape, S – the segmental output tape, and I – the input.

6 Proposed model

As presented in Figure 3, our proposed model, i.e. 2-way 2,2 MT FST, consists of 1-way and 2-way deterministic transducers, which together operate on two tapes. Both input and output are represented on two separate tapes: Tonal and Segmental Tape. Such representation is desired as it correctly mimics linguistic representations of tonal languages, where segments and tones act independently from each other. On the T-tape, a 1-way FST takes the *H* tone associated with the lexical representation of *ndáp* 'house' and outputs *HL*∼ by implementing the Opposite Tone Insertion function. On the S-tape, a 2-way FST takes *ndap* as an input, and outputs a faithful copy of that string (*ndap ndap)*. The ∼ symbol significantly indicates the morpheme boundary and facilitates further output linearization. A detailed derivation of *ndáp* ↦ *ndâp ndap* is shown in Table 4.

Figure 3 also represents two additional 'transformations': splitting and linearizing. First, the phonetic transcription of a string (*ndáp*) is split into tones and segments with a 1,2 MT FST. The output (H, ndap) serves as an input to the 2-way 2,2 MT FST. After the two processes discussed above apply, together acting on both tapes, the output is then linearized with an 2,1 MT FST. The composition of those three machines, i.e. 1,2 MT, 2-way 2,2 MT FST, and 2,1 MT FSTs is particularly useful in applications where a phonetic or orthographic representations needs to be processed.

As was discussed in Section 2, the tone on the second copy is dependent on whether there was an H tone preceding the reduplicated phrase. If there was one, the tone on the reduplicant will be H. Otherwise, the L-Default Insertion rule will insert L tone onto the toneless TBU of the second copy. Because those tonal changes are not part of the reduplicative process *per se*, we do not represent them either in our model in Figure 3, or in the derivation in Table 4. Those alternations could be accounted for with 1-way FST by simply inserting H or L tone to the output of the composed machine represented in Figure 3.

Modelling verbal reduplication and the tonal processes revolving around it (see Table 2) works in the exact same way as described above for nominal reduplication. The only difference are the functions applied to the T-tape.

7 An Alternative to 2-Way Automata

2-way n,m MT FST generalize regular functions (Filiot and Reynier, 2016) to functions from n-strings to m-strings. It is worth asking however, what each of these mechanisms brings, especially in light of fundamental operations such as functional composition.

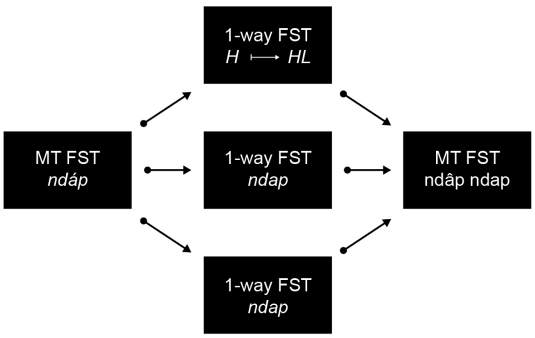

Figure 5: An alternative model for Shupamem reduplication

For instance, it is clear that 2-way $1,1$ MT FSTs can handle full reduplication in contrast to 1-way $1,1$ MT FSTs which cannot. However, full reduplication can also be obtained via the composition of a 1-way 1,2 MT FST with a 1-way $2,1$ MT FST. To illustrate, the former takes a single string as an input, e.g. *ndap*, and 'splits' it into two identical copies represented on two separate output tapes. Then the 2-string output by this machines becomes the input to the next 1-way 2,1 MT FST. Since this machine is asynchronous, it can linearize the

State	Segment-tape	Tone-tape	S-output	T-output
q_0	$\rtimes ndap \ltimes$	$\rtimes H \ltimes$	λ	λ
q_1	$\rtimes ndap \ltimes$ **S**:⋊:+1	$\rtimes H \ltimes$ **T**:⋊:+1	n	HL
q_1	$\rtimes ndap \ltimes$ **S**:n:+1	$\rtimes H \ltimes$ **T**:H:+1	nd	HL∼
q_1	$\rtimes ndap \ltimes$ **S**:d:+1	$\rtimes H \ltimes$ **T**:⋉:0	nda	HL∼
q_1	$\rtimes ndap \ltimes$ **S**:a:+1	$\rtimes H \ltimes$ **T**:⋉:0	ndap	HL∼
q_1	$\rtimes ndap \ltimes$ **S**:p:+1	$\rtimes H \ltimes$ **T**:⋉:0	ndap∼	HL∼
q_2	$\rtimes ndap \ltimes$ **S**:⋉:-1	$\rtimes H \ltimes$ **T**:⋉:0	ndap∼	HL∼
q_2	$\rtimes ndap \ltimes$ **S**:p:-1	$\rtimes H \ltimes$ **T**:⋉:0	ndap∼	HL∼
q_2	$\rtimes ndap \ltimes$ **S**:a:-1	$\rtimes H \ltimes$ **T**:⋉:0	ndap∼	HL∼
q_2	$\rtimes ndap \ltimes$ **S**:d:-1	$\rtimes H \ltimes$ **T**:⋉:0	ndap∼	HL∼
q_2	$\rtimes ndap \ltimes$ **S**:n:-1	$\rtimes H \ltimes$ **T**:⋉:0	ndap∼	HL∼
q_3	$\rtimes ndap \ltimes$ **S**:⋊:+1	$\rtimes H \ltimes$ **T**:⋉:0	ndap∼n	HL∼
q_3	$\rtimes ndap \ltimes$ **S**:n:+1	$\rtimes H \ltimes$ **T**:⋉:0	ndap∼nd	HL∼
q_3	$\rtimes ndap \ltimes$ **S**:d:+1	$\rtimes H \ltimes$ **T**:⋉:0	ndap∼nda	HL∼
q_3	$\rtimes ndap \ltimes$ **S**:a:+1	$\rtimes H \ltimes$ **T**:⋉:0	ndap∼ndap	HL∼
q_3	$\rtimes ndap \ltimes$ **S**:p:+1	$\rtimes H \ltimes$ **T**:⋉:0	ndap∼ndap	HL∼

Table 4: Derivation of *ndáp* ↦ *ndâp ndap* 'houses'
This derivation happens in the two automata in the center of Figure 3. The FST for the Segmental tier is the one shown in Figure 1, and the states in the table above refer to this FST.

2-string (e.g. *(ndap, ndap))* to a 1-string (*ndap ndap*) by reading along one of these input tapes (and writing it) and reading the other one (and writing it) only when the read head on the first input tape reaches the end. Consequently, an alternative way to model full reduplication is to write the exact same output on two separate tapes, and then linearize it. Therefore, instead of implementing Shupamem tonal reduplication with 2-way 2,2 MT FST, we could use the composition of two 1-way MT-FST: 1,3 MT-FST and 3,1 MT-FST as shown in Figure 5. (We need three tapes, two Segmental tapes to allow reduplication as just explained, and one Tonal tape as discussed before.)

This example shows that additional tapes can be understood as serving a similar role to registers in register automata (Alur and Černý, 2011; Alur et al., 2014). Alur and his colleagues have shown that deterministic 1-way transducers with registers are equivalent in expressivity to 2-way deterministic transducers (without registers).

8 Beyond Shupamem

The proposed model is not limited to modeling full reduplication in Shupamem. It can be used for other tonal languages exhibiting this morphological process. We provide examples of the applicability of this model for the three following languages: Adhola, Kikerewe, and Shona. And we predict that other languages could also be accounted for.

All three languages undergo full reduplication at the segmental level. What differs is the tonal pattern governing this process. In Adhola (Kaplan, 2006), the second copy is always represented with a fixed tonal pattern H.HL, where '.' indicates syllable boundary, irregardless of the lexical tone on the non-reduplicated form. In the following examples, unaccented vowels indicate low tone. For instance, *tiju* 'work' ↦ *tija tíjâ* 'work too much', *ʧemó* 'eat' ↦ *ʧemá ʧ↓émâ* 'eat too much'. In Kikerewe (Odden, 1996), if the first (two) syllable(s) of the verb are marked with an H tone, the H tone would also be present in the first two syllables of the reduplicated phrase. On the other hand, if the last two syllables of the non-reduplicated verb are marked with an H tone, the H tone will be present on the last two syllables of the reduplicated phrase. For instance, *bíba* 'plant' ↦ *bíba biba* 'plant carelessly, here and there', *bibílé* 'planted (yesterday)' ↦ *bibile bibílé* 'planted (yesterday) carelessly, here and there'. Finally, in KiHehe (Odden and Odden, 1985), if an H tone appears in the first syllable of the verb, the H tone will also be present in the first syllable of the second copy, for example *dóongoleesa* 'roll' ↦ *dongolesa dóongoleesa* 'roll a bit'.

The above discussed examples can be modelled in a similar to Shupamem way, such that, first,

the input will be output on two tapes: Tonal and Segmental, then some (morpho-)phonological processes will apply on both level. The final step is the 'linearization', which will be independent of the case. For example, in Kikerewe, if the first tone that is read on the Tonal tape is H, and a vowel is read on the Segmental tape, the output will be a vowel with an acute accent. If the second tone is L, as in *bíba*, this L tone will be 'attached' to every remaining vowel in the reduplicated phrase. While Kikerewe provides an example where there are more TBUs than tones, Adhola presents the reverse situation, where there are more tones than TBU (contour tones). Consequently, it is crucial to mark syllable boundaries, such that only when '.' or the right edge marker ($\rtimes$) is read, the FST will output the 'linearized' element.

9 Conclusion

In this paper we proposed a deterministic finite-state model of total reduplication in Shupamem. As it is typical for Bantu languages, Shupamem is a tonal language in which phonological processes operating on a segmental level differ from those on suprasegmental (tonal) level. Consequently, Shupamem introduces two challenges for 1-way FSTs: language copying and autosegmental representation. We addressed those challenges by proposing a synthesis of a deterministic 2-way FST, which correctly models total reduplication, and a MT FST, which enables autosegmental representation. Such a machine operates on two tapes (Tonal and Segmental), which faithfully replicate the linguistic analysis of Shupamem reduplication discussed in Markowska (2020). Finally, the outputs of the 2-way 2,2 MT FST is linearized with a separate 2,1 MT FST outputting the desired surface representation of a reduplicated word. The proposed model is based on previously studied finite-state models for reduplication (Dolatian and Heinz, 2020) and tonal processes (Dolatian and Rawski, 2020b,a).

There are some areas of future research that we plan to pursue. First, we have suggested that we can handle reduplication using the composition of 1-way deterministic MT FSTs, dispensing with the need for 2-way automata altogether. Further formal comparison of these two approaches is warranted. More generally, we plan to investigate the closure properties of classes of 2-way MT FSTs. A third line of research is to collect more examples of full reduplication in tonal languages and to include them in the RedTyp database (Dolatian and Heinz, 2019) so a broader empirical typology can be studied with respect to the formal properties of these machines.

References

Rajeev Alur, Adam Freilich, and Mukund Raghothaman. 2014. Regular combinators for string transformations. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, CSL-LICS '14, pages 9:1–9:10, New York, NY, USA. ACM.

Rajeev Alur and Pavol Černý. 2011. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '11, page 599–610, New York, NY, USA. Association for Computing Machinery.

Steven Bird. 2001. Orthography and identity in Cameroon. *Written Language & Literacy*, 4(2):131–162.

Jane Chandlee. 2017. Computational locality in morphological maps. *Morphology*, pages 1–43.

Jane Chandlee and Jeffrey Heinz. 2012. Bounded copying is subsequential: Implications for metathesis and reduplication. In *Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology*, pages 42–51, Montréal, Canada. Association for Computational Linguistics.

Hossep Dolatian and Jeffrey Heinz. 2019. Redtyp: A database of reduplication with computational models. In *Proceedings of the Society for Computation in Linguistics*, volume 2. Article 3.

Hossep Dolatian and Jeffrey Heinz. 2020. Computing and classifying reduplication with 2-way finite-state transducers. *Journal of Language Modelling*, 8(1):179–250.

Hossep Dolatian and Jonathan Rawski. 2020a. Computational locality in nonlinear morphophonology. Ms., Stony Brook University.

Hossep Dolatian and Jonathan Rawski. 2020b. Multi input strictly local functions for templatic morphology. In *In Proceedings of the Society for Computation in Linguistics*, volume 3.

David M. Eberhard, Gary F. Simmons, and Charles D. Fenning. 2021. *Enthologue: Languages of the World*. 24th edition. Dallas, Texas: SIL International.

Emmanuel Filiot and Pierre-Alain Reynier. 2016. Transducers, logic and algebra for functions of finite words. *ACM SIGLOG News*, 3(3):4–19.

Patric C. Fischer. 1965. Multi-tape and infinite-state automata-a survey. *Communications of the ACM*, pages 799–805.

John Goldsmith. 1976. *Autosegmental Phonology*. Ph.D. thesis, Massachusetts Institute of Technology.

Nizar Habash and Owen Rambow. 2006. Magead: A morphological analyzer for Arabic and its dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (Coling-ACL'06)*, Sydney, Australia.

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

Larry Hyman. 2014. How autosegmental is phonology? *The Linguistic Review*, 31(2):363–400.

Adam Jardine. 2016. Computationally, tone is different. *Phonology*, 33:247–283.

Aaron F. Kaplan. 2006. Tonal and morphological identity in reduplication. In *Proceedings of the Annual Meeting of the Berkeley Linguistics Society*, volume 31.

George Anton Kiraz. 2000. Multi-tiered nonlinear morphology using multi-tape finite automata: A case study on Syriac and Arabic. *Computational Linguistics*, 26(1):77–105.

William R. Leben. 1973. *Suprasegmental Phonology*. Ph.D. thesis, Massachusetts Institute of Technology.

Magdalena Markowska. 2019. Tones in Shuapmem possessives. Ms., Graduate Center, City University of New York.

Magdalena Markowska. 2020. Tones in Shupamem reduplication. *CUNY Academic Works*.

John McCarthy. 1981. A prosodic theory of nonconcatenative morphology. *Linguistic Inquiry*, 12:373–418.

Abdoulaye L. Nchare. 2012. *The Grammar of Shupamem*. Ph.D. thesis, New York University.

David Odden. 1996. Patterns of reduplication in kikerewe. *OSU WPL*, 48:111–148.

David Odden and Mary Odden. 1985. Ordered reduplication in KiHehe. *Linguistic Inquiry*, 16:497–503.

Michael O Rabin and Dana Scott. 1959. Finite automata and their decision problems. *IBM journal of research and development*, 3:114–125.

Jonathan Rawski and Hossep Dolatian. 2020. Multi-input strict local functions for tonal phonology. *Proceedings of the Society for Computation in Linguistics*, 3(1):245–260.

Brian Roark and Richard Sproat. 2007. *Computational Approaches to Morphology and Syntax*. Oxford University Press, Oxford.

Carl Rubino. 2013. *Reduplication*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Bruce Wiebe. 1992. Modelling autosegmental phonology with multi-tape finite state transducers.

Edwin S. Williams. 1976. Underlying tone in Margi and Igbo. *Linguistic Inquiry*, 7:463–484.