XSMA: A Finite-state Morphological Analyzer for Swahili

Jonathan Lipps

MPhil Essay

General Linguistics and Comparative Philology

Oxford University

Trinity Term 2011

Contents

The text of this thesis contains 7,485 words, including footnotes and figures.

1 Introduction

Swahili is a Bantu language of East Africa, recognized as one of the national languages of Tanzania and Kenya (Russell and Perrott 1996:vii). Like other Bantu languages, Swahili makes heavy use of agglutinative morphology, especially in verbs. This paper will outline a computational model of Swahili morphology, originally developed in order to support a syntactic model of the same language. The abbreviated name for the morphological analysis system described in this paper is XSMA: an XFST-based Swahili Morphological Analyzer.

In the following sections, I will give an overview of the basic morphological facts of Swahili, and then move on to discuss the theoretical foundations of finite-state-based morphological analysis. It will then be possible to show how XFST (a system for compiling finite state transducers) can be used to model the particular morphological facts of Swahili. I will also discuss issues, linguistic and otherwise, raised in the modeling process. While XSMA does not currently exhibit complete coverage of the morphological structure of Swahili, I hope to show that it could be extended in powerful ways, and that its development is in any case an interesting exercise in representing some of the most important facets of this morphologically-rich language.

2 Swahili Morphology

Almost every word in a Swahili sentence bears some kind of morphological marking, owing primarily to the language's noun class system. Scholars recognize 18 noun classes (which can without too much harm be thought of as analogous to gender) whose members share certain prefixes. To form grammatical utterances, Swahili nouns are put into class agreement relationships with modifying adjectives and verbs; class prefixes therefore make a frequent appearance in a typical sentence.

There is also complex morphological structure in verbs, which are essentially templates filled with various morphological markers, each of which contributes certain information. In this section, I will detail the aspects of Swahili morphology I take to be most relevant to building a computational analyzer of it. Interactions with phonology will, of necessity, also come into view.

2.1 Noun Classes

It is not entirely straightforward to count the noun classes of Swahili. Linguistically-focused grammars (like Mohammed (2001)) and scholarly articles currently assume that there are 18 such classes, whose distribution merits a few remarks:

- Classes 1 - 14 consist of 7 singular and plural pairs of nouns which belong to those classes lexically. For example, the Swahili word for 'chair' is class 7 *kiti*, and 'chairs' is class 8 *viti*. Nouns may then be thought of as really belonging to e.g. class-pair '1/2' or '3/4', along with a specification of number. The question of how best to gloss a given noun depends on which we think exhibit greater linguistic reality: classes or class-pairs. In this paper, again following convention and to reduce the length of glosses, I will follow the first strategy and gloss *kiti* as '7.chair'.

- Classes 15 - 18 are 'special-purpose' classes; of the four, only class 16 has any nouns lexically defined to belong it. Class 15 is used for infinitive constructions, and classes

16 - 18 are used primarily in morphological marking of location (which fact explains the one class 16 noun, *pahali* 'place').

- From a lexical perspective, nouns belong to only one class-pair, but may sometimes be used with the morphological marking of a different class-pair (depending on the resulting class, some additional sense is conveyed, e.g., *mtu* '1.person' → *jitu* '5.giant.person').

In terms of morphology, there are two sets of class prefixes. Prefixes from the first set (the *nominal class prefixes*) are found on nouns themselves and on open-class adjectives. Morphemes from the second set (the *verbal class prefixes*) appear on verbs as well as on a special closed-class set of adjectives. These different prefixes will be listed and explained in the following sections.

2.2 Nominal and Adjectival Morphology

Swahili nouns consist of a nominal class prefix and a stem. In some cases, stems appear in multiple class-pairs, suggesting that at some stage in the language's history stems were the basic lexical items, and could productively be associated with one or more of the classes to form a complete nominal concept. This situation no longer obtains in a completely productive fashion, however.

Since nouns, for the most part, are used according to their lexically-defined class-pair, I will gloss *mkulima* 'farmer' as in (1) and not (2), making clear that the stem *-kulima* is semantically opaque on its own:[1,2]

(1) *mkulima*
1.farmer
'farmer'

[1]The converse will hold of adjectives, whose stems combine regularly with all class prefixes.

[2]The glosses in this paper are formatted according to the Leipzig Glossing Rules (Bickel et al. 2008). Two aspects of my glossing strategy differ from the Leipzig rules: firstly, numerals standing alone (e.g. '10') refer to classes, not persons. Secondly, I have introduced the abbreviation 'AN' to designate animate status.

(2) *m-kulima*
1-farmer
'person'

Having at least mentioned some of the conceptual wrinkles in describing noun classes, we are able to lay out the basic list of nominal class prefixes, shown in Table 1:

Class	**Prefix**	**Example**	**Gloss**
1	m-	mkulima	farmer
2	wa-	wakulima	farmers
3	m-	mto	river
4	mi-	mito	rivers
5	(ji-)	jicho	eye
6	ma-	macho	eyes
7	ki-	kitu	thing
8	vi-	vitu	things
9	(n-)	ndoto	dream
10	(n-)	ndoto	dreams
11	u-	ufagio	broom
12	(n-)	fagio	brooms
13	u-	ubaya	badness
14	ma-	mabaya	bad actions
15	ku-	—	
16	pa-	pahali	place
17	ku-	—	
18	mu-	—	

Table 1: Basic nominal class prefixes

As is clear from looking at the table, the nominal class prefixes are not all unique—it is only when a noun is shown in both the singular and plural that its class memberships can be unambiguously determined. These nominal class prefixes appear on nouns themselves, and also on adjectives, where they serve to enforce concordial agreement. Example (3) shows an adjective in proper concord with its noun, while (4) emphasizes the ungrammaticality of non-agreement:

(3) *mawe ma-zito*
6.stone 6-heavy
'heavy stones'

(4) **miti ma-refu*
4.tree 6-tall
'tall trees'

These basic and clear patterns of nominal and adjectival morphology are complicated by phonology; there are many phonological environments in which the nominal class prefixes or stems are altered, sometimes radically. In (5), for example, the class 12 nominal prefix shows up as *m-* and then *ny-*, neither of which are the expected *n-*![3]

(5) *mbao ny-ekundu*
12.plank 12-red
'red planks'

2.3 Verbal Morphology

Swahili verbs can be viewed as a morphological template, in which morphemes with appropriate meanings are placed. In this paper, I loosely follow Schadeberg (1984) and Edelsten et al. (2010) in describing the template as in Table 2 (p. 8). These slots only constitute a range of possibilities for building grammatical verbs; apart from slots 6b and 7, in fact, they are all optional.

2.3.1 Class Markers

Given that verbal morphology is such an important part of the overall morphological system of Swahili, I will give a brief overview of some morphemes which may be used in slots of the verbal template. Of primary importance are the verbal class prefixes, shown in Table 3 (p.

[3] I will describe the phonological motivations for such apparent irregularities in section 2.4.

Slot	Morpheme Description
1	Negative Marker 1
2	Subject Marker
3	Negative Marker 2
4a	Tense Marker
4b	Tensed Relative Marker
5	Object Marker
6a	Monosyllabic verb extension
6b	Verbal Base
7	Final Vowel
8a	General Relative Marker
8b	Plural Imperative Marker

Table 2: Verbal morpheme slots

9). These can be used as subject and object markers (in slots 2 and 5, respectively) whose main function is to exhibit class agreement with the (either overt or elliptical) subject and object.

One detail of these prefixes does need to be discussed: in Swahili, the animacy of a noun greatly influences which verbal class prefixes are used. Regardless of the lexical class of a noun, if it is animate, it will agree with class-pair 1/2 subject and object markers. These markers are categorized for person, as Table 4 (p. 10) details.

Subject and object markers can be used without overt antecedents, but in doing so restrict the class of possible intended referents to those grammatically or semantically compatible with their class:

(6) ***A****-na-lal-a*
AN.3SG-PRES-sleep-FV
'(S)he is sleeping.'

(7) *Ni-me-****vi****-som-a*
AN.1SG-PERF-8.OBJ-read-FV
'I have read them (class 8 things, maybe books?)'

Class	Subject Marker	Object Marker
1	ni-,u-,a-	ni-,ku-,m-
2	tu-,m-,wa-	tu-,wa-,wa-
3	u-	u-
4	i-	i-
5	li-	li-
6	ya-	ya-
7	ki-	ki-
8	vi-	vi-
9	i-	i-
10	zi-	zi-
11	u-	u-
12	zi-	zi-
13	u-	u-
14	ya-	ya-
15	ku-	—
16	pa-	po-
17	ku-	ko-
18	mu-	mo-

Table 3: Verbal class prefixes as subject and object markers

The verbal class prefixes are not used solely in verbal morphology; there is a closed class of adjectives (including demonstratives and possessives) which show class agreement by use of the verbal rather than the nominal class prefixes.

2.3.2 Negative Markers

In the verbal 'template' described in Table 2, there are two slots for negative morphemes. The negative prefix found in slot 1 is *ha-*.[4] The other negative marker *-si-*, found in slot 3, is used only in certain constructions like the subjunctive, or with certain tense markers. It cannot co-occur with a negative morpheme in slot 1. (8) and (9) show typical uses of these two negative morphemes:

[4]With animate verbal prefixes (see again Table 4), there is some irregularity: we do not find *ha-ni-* or *ha-u-*, for example, but the rather more synthetic *si-* and *hu-*, respectively.

Person	Number	Subject Marker	Object Marker	Gloss
1	Sg	ni-	-ni-	AN.1SG
2	Sg	u-	-ku-	AN.2SG
3	Sg	a-	-m-	AN.3SG
1	Pl	tu-	-tu-	AN.1PL
2	Pl	m-	-wa-	AN.2PL
3	Pl	wa-	-wa-	AN.3PL

Table 4: Subject and object verb markers for agreement with animate nouns

(8) *Si-ta-ondok-a*
AN.1SG.NEG-FUT-leave-FV
'I won't leave.'

(9) *U-si-ondok-e*
AN.2SG-NEG-leave-FV
'You shouldn't leave' or 'Don't leave'.

2.3.3 Tense Markers

Slot 4a in the verbal template is filled by so-called 'tense' markers; in fact, the morphemes in this category can convey more than tense information, including mood, aspect, and discourse relationships. Table 5 (p. 11) lists the morphemes available to the Swahili speaker.

2.3.4 Relative Markers

Interestingly, Swahili can form relative clauses by including certain morphemes in the verb template. I call this clause-formation strategy the 'verbal affix strategy', and it is signalled by inserting a relative marker into verbal morpheme slot 4b or 8a. The relative marker must agree in class or animacy with the head noun. Example (10) shows how the animate 3SG relative marker *-ye-* allows an entire relative clause to be built from one verb:

Morpheme	Basic meaning	Notes
-na-	Present tense	
-li-	Past tense	
-ku-	Past tense	Allomorph of *-li-* in negative contexts
-ta-	Future tense	
-taka-	Future tense	Allomorph of *-ta-*
-me-	Present perfect	
-ja-	Present perfect	Allomorph of *-me-* in negative contexts
-nge-	Present conditional	
-ngali-	Past conditional	
-ki-	Suppositional	
-sipo-	Negative suppositional	
-ki-	Present/future imperfect	
-ka-	Consecutive	
-hu-	General	

Table 5: 'Tense' markers used in verb slot 4a

(10) *mtu* *a-li-ye-kw-end-a*
1.person AN.3SG.SBJ-PAST-AN.3SG.REL-SE-go-FV
'person who went'

2.3.5 The Stem Extension *-ku-*

Verb slot 6 is optionlly filled by the monosyllabic verb stem extension *-ku-* (sometimes realized as *-kw-*, and in either case glossed as SE). This morpheme does not contribute to the meaning of the verb, but is rather present only as a prefix to monosyllabic verb stems. Because of the rules of Swahili stress, a present-tense verb based on the stem *-l-* 'eat' is not allowed:

(11) **Ni-**na**-l-a*
AN.1SG-PRES-eat-FV
'I am eating.'

In this construction, *-na-* would receive the main stress (which in Swahili words is always on the penult); however, *-na-* is of a class of Swahili morphemes which are prohibited from

taking such stress, and therefore *-ku-* is epenthesized:

(12) *Ni-na-**ku**-l-a*
AN.1SG-PRES-SE-eat-FV
'I am eating.'

2.3.6 The Verbal Base and Final Vowel

There are two main categories of verbs in Swahili:

1. Bantu-stem verbs arose in the natural history of Swahili itself, or have been imported and naturalized. In dictionaries, these verbs are listed as stems along with the final vowel *-a*, though this vowel can change in some circumstances (to *-i*, for example, in the present negative, or to *-e* in the subjunctive).

2. Arabic-stem verbs were imported, and have invariable final vowels (*-i*, *-e*, or *-u*). This invariability is relevant in constructions where the change of final vowel is significant.

Compare, for example, two negative subjunctive constructions (used here as negative imperatives). In (13), the verb base *-saha-u* 'forget' is of Arabic origin, and its final vowel does not change to *-e*, as it does with *-lal-a* 'sleep':

(13) *U-si-saha-u!*
AN.2SG-NEG-forget-FV
'Don't forget!'

(14) *U-si-lal-e!*
AN.2SG-NEG-sleep-FV
'Don't sleep!'

The verbal base can itself be modified by rules of derivational morphology. One rule, for example, forms a passive verb base by appending *-w-* to the stem, as in *-pend-a* 'love' → *-pend-w-a* 'be loved'.

2.3.7 The Plural Imperative Marker

In a small number of constructions, for example imperatives and subjunctives, a special plural marker *-ni* is added in the final slot of the verbal template, which attributes plural number to the addressee, as in (15):

(15) *Sikiliz-e-ni!*
listen-FV-PL
'(You all) listen!'

2.3.8 Verb Template Examples

I conclude this overview of Swahili verbal morphology by giving in tabular form a set of examples of how the verb template can be instantiated in a variety of ways:

1	**2**	**3**	**4a**	**4**	**5**	**6a**	**6b**	**7**	**8**	**8b**	**Translation**
	u-		li-				anguk-	a			'You fell down'
	2SG-		PAST-				fall	-FV			
	wa-		taka-	o-		ku-	l	-a			'Those who will eat'
NEG-	3PL-		FUT-	3PL.REL-		SE-	eat	-FV			
	m-	si-			tu-		pig	-e		-ni	'You all should not beat us'
	2PL-	NEG-			1PL-		beat	-FV		-PL	
	ni-				zi-		pik	-a	-zo		'Things which I usually cook'
	1SG-				10-		cook	-FV	10.REL		
	u-						l	-e			'You should eat'
	2SG-						eat	-FV			
ha-	zi-		ku-				pikw	-a			'They were not cooked'
NEG-	10-		PAST-				be.cooked-	FV			
							tok	-a			'Leave!'
							leave	-FV			

Table 6: Verbal template examples

2.4 Phonological Rules

Forming morphologically-valid Swahili words is not always as easy as simply stringing morphemes together. Class prefixes, for example, are subject to various transformations when they appear in certain phonological environments. The effect of these transformations is obvious in looking at lists of Swahili nouns which belong to the same class-pair. Table 7 shows the underlying and surface forms for a number of nouns which illustrate this point.

Class	Prefix	Nominal base	Surface form	Meaning
1	m-	-alimu	mwalimu	teacher
2	wa-	-alimu	walimu	teachers
3	m-	-aka	mwaka	year
		-wa	muwa	sugar-cane
5	ji-	-ino	jino	tooth
6	ma-	-ino	meno	teeth
7	ki-	-umba	chumba	room
8	vi-	-umba	vyumba	rooms
9/10	n-	-umba	nyumba	house(s)
		-vua	mvua	rain(s)
		-limi	ndimi	tongue(s)
		-jia	njia	road(s)
11	u-	-avu	wavu	net
12	n-	-avu	nyavu	nets
		-bao	mbao	boards
13	u-	-ali	wali	rice
15	ku-	-enda	kwenda	going

Table 7: Underlying and surface forms of some nouns

A few observations can be made at this stage.[5] First, the classes which share prefixes are involved in the same phonological changes (cf. classes 9, 10, and 12). Second, long vowels formed as a result of morpheme concatenation are frequently shortened. Third, [u] → [w] in many contexts, (thus [mwa.ka] 'year', not three-syllable [mu.a.ka]). It is also clear from the

[5]Since Swahili orthography is generally an excellent guide to pronunciation, I will often not give phonetic transcriptions of words or morphemes. The orthographic elements which differ significantly from the IPA equivalents are as follows: /ch/ = [tʃ], /y/ = [j], /j/ = [dʒ] or [ɟ], /sh/ = [ʃ], /ny/ = [ɲ], /ng/ = [ŋg], /ng'/ = [ŋ], /th/ = [θ], /dh/ = [ð], and /gh/ = [ɣ].

table that there is a system of contraction in Swahili, e.g., as in class 6 *meno* 'teeth' < *ma-ino*. These and other rules are mirrored in the formation of adjectives; most adjectives take the nominal class prefixes in order to agree with the nouns they modify, and thus participate in the same phonological operations. Table 8 shows how several adjectives incorporate the class prefixes.

Class	**Prefix**	**-baya** 'small'	**-ema** 'good'	**-refu** 'tall'
1	m-	mbaya	mwema	mrefu
2	wa-	wabaya	wema	warefu
3	m-	mbaya	mwema	mrefu
4	mi-	mibaya	myema	mirefu
5	(ji-)	baya	jema	refu
6	ma-	mabaya	mema	marefu
7	ki-	kibaya	chema	kirefu
8	vi-	vibaya	vyema	virefu
9	n-	mbaya	njema	ndefu
10	n-	mbaya	njema	ndefu
11	m-	mbaya	mwema	mrefu
12	n-	mbaya	njema	ndefu
13	m-	mbaya	mwema	mrefu
14	ma-	mabaya	mema	marefu

Table 8: Examples of phonological change in adjectives

Of particular interest in these tables is what happens in classes 9, 10, and 12, where I have listed the prefix as *n-*. Some of the apparently random behavior can be explained by assuming that the underlying morpheme is actually *ni-* (which according to Russell and Perrott (1996) is a plausible historical reconstruction). The following set of phonological rules show how that assumption adequately covers the data.[6,7]

[6]In all of the rules, I have used $]_\mu$ to refer to the morpheme boundary, since we cannot assume that these rules apply to just any phonological environment.

[7]I agree with Mpiranya (1995) that writing synchronic phonological rules is an unhelpful way to analyze the variation in the class 9/10/12 prefix. Diachrony provides a better linguistic explanation, and the rules as I have laid them out are a clunky way to capture the facts; however, XSMA needs 'synchronic' formulations in order to function well.

(16) $[\ +\text{SYLL}\ ] \longrightarrow \emptyset \ / \ \underline{\quad}\ ]_\mu \ \ \text{C}$

"[i] in *ni-* is deleted before a consonant."

(17) $[\ +\text{SYLL}\ ] \longrightarrow [\ -\text{SYLL}\ ] \ / \ \underline{\quad}\ ]_\mu \ \ \text{V}$

"[i] in *ni-* is converted to a glide before a vowel."

(18) $[\ +\text{NASAL}\ ] \longrightarrow \emptyset \ / \ \underline{\quad}\ ]_\mu \ \ \Big[[[\ -\text{VOICED}\ ]\ldots\ ]_\sigma\ \sigma+\Big]_\omega$

"[n] in *ni-* is deleted before an unvoiced consonant when the stem is not monosyllabic."

(19) $[\ +\text{NASAL}\ ] \longrightarrow [\ \alpha\text{LABIAL}\ ] \ / \ [\ \underline{\quad}\ ]_\mu \ \ [\ \alpha\text{LABIAL}\ ]$

"Any remaining [n] takes on the labiality of the initial stem consonant. I.e., progressive assimilation takes place."

(20) $[\ +\text{APPROX}\ ] \longrightarrow \begin{bmatrix} -\text{APPROX} \\ +\text{CONSONANT} \end{bmatrix} \ / \ [\ +\text{NASAL}\ ]\ ]_\mu \ \underline{\quad}$

"If the first vowel of the stem is an approximant, it becomes an obstruent with the same place features."

We could write similar rules for changes, e.g. the palatalization of *ki-* in class-7-agreeing adjectives like *chema* [tʃema] < *ki-ema*; moreover, the verbal class prefixes are subject to sound changes of their own. For reasons of space I will leave out such analyses, but will return to them when discussing how XSMA models these rules in order to parse words into the correct morphemes.

3 Finite-State Transducers

Finite-state transducers (FSTs) represent a class of *finite-state networks* (FSNs; equivalently *finite-state automata* or *machines*). FSNs consist of *states* (represented as labelled nodes) and *transitions* (represented as labelled arcs which can be thought of as relations defining accessibility between states). FSNs are powerful tools for modelling real-world phenomena describable in terms of changes of state and the corresponding actions which cause (or accompany) a particular change of state; there are intuitively many such phenomena for which FSNs are explanatory representations. One simple mechanical example from Beesley and Karttunen (2003) is a cola machine. Such a machine starts in a state where it awaits some coin input, and delivers a cola when and only when a total of 25 cents have been inserted into the machine. The FSN in Figure 1 shows the various ways that 5-, 10-, and 25-cent coins can be inserted in order to place the machine in the end state of delivering a cola.

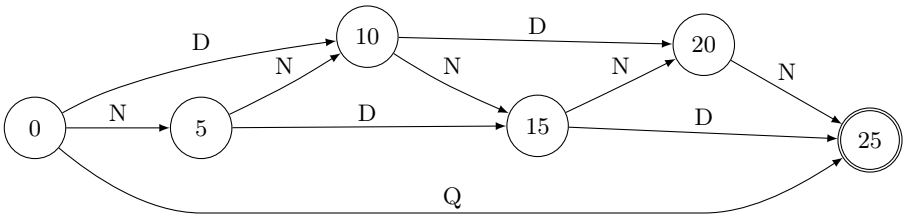


Figure 1: FSN model of a cola machine

In this machine, the traversing of an arc represents the *consumption*, from the perspective of the machine, of a coin. It is only when the proper coin is made available for consumption that the arc may be traversed and the attached state entered. Cola machines are not the only systems amenable to this kind of FSN analysis—in section 3.2 I will make explicit how a morphological analysis in particular can be produced by transitions through a sequence of states wherein the string to be analyzed is consumed by those transitions.

3.1 Properties of FSNs and FSTs

Before I discuss how natural language in particular can be modelled with FSTs, I will make a few general points about the properties of FSNs and FSTs. FSNs can be used not just to consume inputs which drive state transitions—they can also be instructed to give output along the way. Such an FSN is called a *transducer* because in moving through transitions it converts an input source (a stream of tokens) into output.

The concept of transduction is a powerful one; we can, for example, construct FSTs whose purpose is to translate words from English to French, as in Figure 2 (Beesley and Karttunen 2003:29). In this example, the consumed token is written on the upper side of the transition, and the produced token is written on the lower side. The symbol ϵ stands for the empty string, and is important since it allows transitions that consume nothing while producing something, or vice versa.

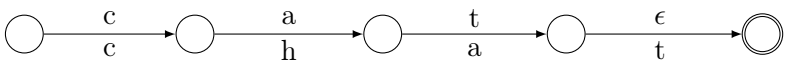


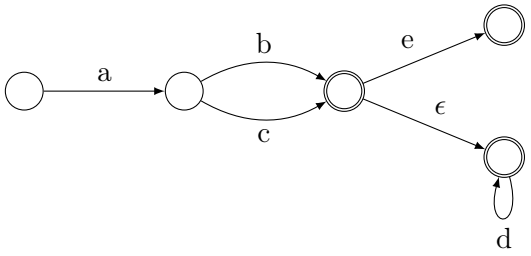
Figure 2: A simple lexical transducer

This transducer accepts the English word 'cat' and produces the French equivalent 'chat' (though it could be inverted to translate in the other direction). FSTs like this one encode a potentially-complicated relation between two strings, and it is this insight which makes FSTs ideal for morphological analysis.

Conveniently for the computational linguist, FSNs and FSTs can in general be equivalently represented as *regular expressions*, special-purpose strings which encode the concepts of states and transitions.[8] For any regular expression, it is possible to show its FSN equivalent, as in (21):

[8]It is not within the scope of this paper to introduce the language of regular expressions; for a full discussion of the subject, see Beesley and Karttunen (2003).

(21) `a [b | c] ([d* | e])`



Regular expressions can also encode FSTs, like the 'cat'-translator. In accord with convention, I will signify the difference in a regular expression between the input symbol and the output symbol by separation with ':', as in `[ a:A ]`, which refers to the consumption of `a` and the simultaneous production of `A`.[9] The regular-expression equivalent of the 'cat'-translator would then be: `[ c:c a:h t:a ϵ:t ]`.[10]

The XFST language provides a number of specialized symbols in order to provide specific support for FSTs over FSNs, and defines a number of advanced syntactic constructions which make the job of stating morphological facts easier. The 'cat'-translator could, in XFST, be equivalently constructed using the more clearly written regular expression `[ {cat}:{chat} ]`.

3.2 An FST-based Approach to Morphological Analysis

Transduction is a powerful way to model morphological analyses. The goal of such an analysis is to consume an input string (a representation of a word in a natural language) and produce an output string (usually a combination of some 'dictionary' form of the word, along with any morphological properties). By convention, these morphological properties are multi-character symbols ('tags': in my implementation, I have chosen to surround tags with the symbol '+').

[9] In the XFST language, ':' is a binary infix operator whose technical function is to relate two symbols by transduction: the symbol or group of symbols on the left are defined to be on the upper side of an arc, and the ones on the right are defined to be on the lower side of the arc.

[10] In the XFST implementation which undergirds XSMA, the ASCII symbol `0` (the numeral zero) denotes the empty string.

Example (22) shows an analysis for the Swahili word *nimezipika*, glossed in (23):[11]

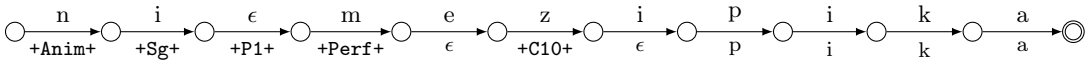
(22) `nimezipika` ⟶ `+Anim+ +Sg+ +P1+ +Perf+ +C10+ pika`

(23) *ni-me-zi-pik-a*
AN.1SG-PERF-10.OBJ-cook-FV
'I have cooked them'

The analysis produced by the transducer is essentially a linguist's gloss of the word, which can in turn be used by morphologically-sensitive applications. Swahili as a language is well-suited to a transduction analysis, since very often the morphemes are separable and can be transduced directly into their tags.

When developing lexical resources in XSMA, I chose to work with input strings (i.e., Swahili words) on the left (or upper) side, and the transduced analysis strings on the right (or lower) side. For some operations,like the application of conditional replacement, e.g. as in phonological rules, it is most straightforward to put the replacement language (the Swahili words) on the lower side. Fortunately, it is very easy to invert the FST in order to accommodate both of these requirements. Example (24) shows the regular expression which encodes the analysis of *nimezipika*, along with its FST representation.

(24) `{ni}:["+Anim+""+Sg+""+P1+"] {me}:"+Perf+" {zi}:"+C10+" p i k a`



Of course, given that the empty string ϵ can be inserted anywhere, on either the upper or lower side of the arc, it is true that any two strings can be related by transduction in an essentially stipulative fashion (by adding ϵ as necessary on either side in order to create the relation). *A fortiori*, any particular Swahili word can be related to its stipulated analysis;

[11] The differences between the analysis string and the gloss in (23) are superficial; see 5.1 for a discussion of why these analysis tags were chosen, rather than, say, `+Anim+ +1SG+ +Perf+ +10OBJ+ pika`.

however, a system where each word is simply part of a stipulated relation with the right morphological tags intuitively does not constitute a real analysis of Swahili morphology.

The goal of building an FST-based morphological analyzer, then, is to construct a network which accepts all and only the valid words of a language, and delivers the right analysis (as defined by the linguist) for each; moreover, it should avoid as far as possible the brute-force enumeration of complex forms. Such a network will naturally be complex, and producing a visual representation of it as an FST would prove an unhelpful way to explore its behavior. Fortunately, we can describe such networks using regular expressions, which a finite-state compiler like XFST can translate into a compact binary version, ready for use in practical morphological analysis.

3.3 The XFST Framework

XFST is a programming language designed by Xerox at the Palo Alto Research Center, and comes bundled with a set of tools for compiling and working with FSTs. In this section, I simply want to mention without explication two features of XFST which go beyond the regular expression language and into the realm of more complicated finite-state operations—*composition* and *replacement* (see Beesley and Karttunen (2003) for more details):

- Composition is an operation which merges the networks described by two regular expressions into one. In XFST, the composition operator is `.o.`; (25) shows the composition of English-French and French-German cat transducers (along with the resulting English-German one):

  (25) `[ {cat}:{chat} ].o.[ {chat}:{Katze} ]` ⇒ `[ {cat}:{Katze} ]`

- Replacement is an operation which takes one network and produces another, possibly with a few differences. XSMA primarily uses *conditional replacement*; these replacements, in both form and function, are analogous to standard phonological rewrite rules,

and were in fact originally modeled on them (Beesley and Karttunen (2003:66)). The syntax for conditional replacement is shown in (26), which says says, "If I see an 'h' in between a 'c' and an 'at', delete it!" (the special symbols `->`, `||`, and `_` separate the different components of the replacement rule):

(26) `[ h -> 0 || c _ a t ]`

4 Analyzing Swahili Morphology with XFST

The XSMA system is constructed entirely in XFST, although open-class lexical stems are stored in Toolbox (Robinson et al. 2007) and exported as XFST regular expressions via a series of Python scripts. The morphological analysis provided by XSMA can be used as input to, *inter alia*, a syntactic parser like XLE (Crouch et al. (2007)), which also requires its own lexical database (hence the use of Toolbox and various scripts to reduce data redundancy). This section is devoted to a detailed overview of the system's components.

4.1 The Basic Architecture

A schematic map of the XSMA architecture is given in Figure 3 (p. 24); it highlights the sub-systems which are combined in order to produce the analyses.

The map is divided (by the dashed line) into the two main parts of the system: the lexicon and the rules. The lexicon is essentially a huge collection of possible Swahili word forms, built by disjunction (denoted by the circular arrowheads) from various sublexicons. The set of rules likewise consists of various bits of phonological or morphological logic, which are joined together via the composition operation (denoted by the diamond-shaped arrowheads). The lexicon and rules are finally composed into the final XSMA network.

The sections of the framework which have been outlined with dotted lines are those which are automatically generated from the Toolbox database; thus, it is the solid-bordered, white-backgrounded boxes which designate areas of active development. In the next few sections, I will detail more specifically how the most important of these components operate.

4.2 Transducer and Symbol Conventions

As explained in earlier sections, the analysis strings consist of morphological tags. By convention, I enclose morphological tags in a pair of '+' symbols so that they are easily distinguished from other elements. Some of the tags of XSMA are listed below:

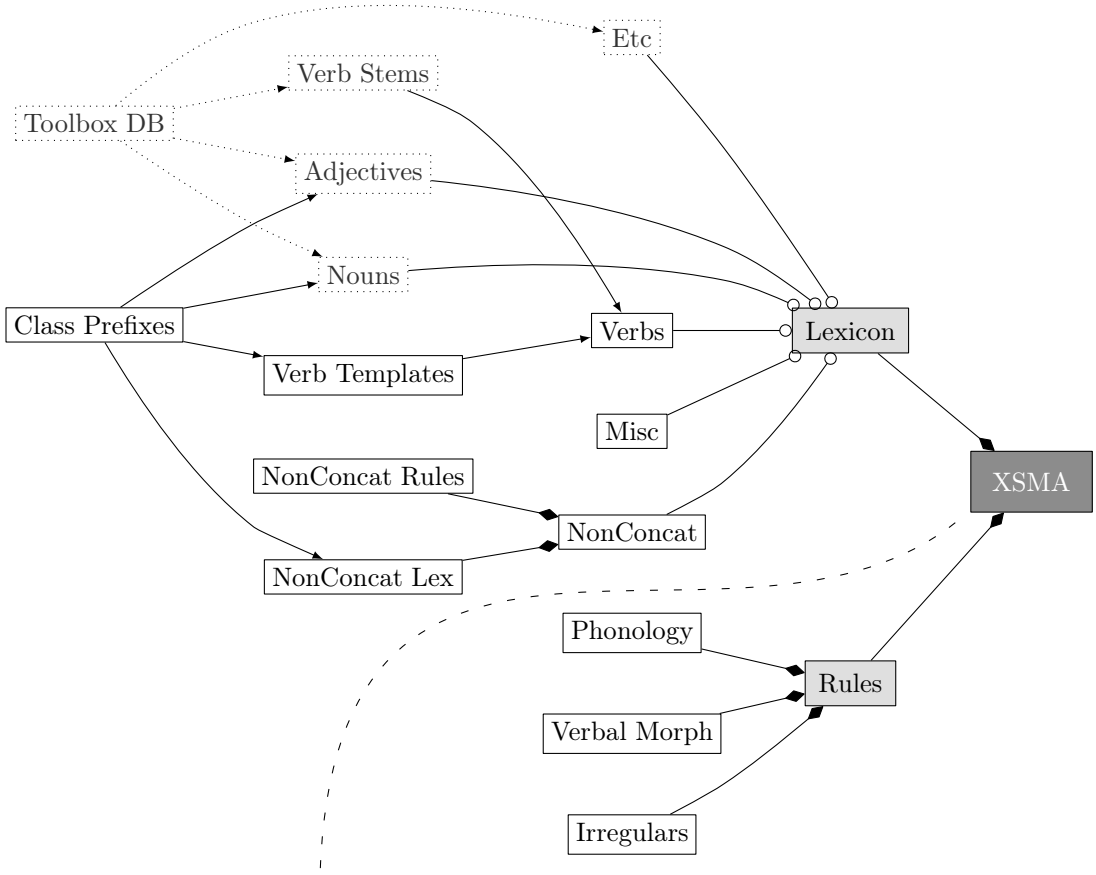


Figure 3: A schematic representation of the XSMA architecture

- `+C1+`: class 1
- `+Pres+`: present tense
- `+Imper+`: imperative mood

It is also quite useful to use special markers in constructing the 'input' language (i.e., regular expressions representing words in a Swahili lexicon, which are used in building the transducer), for instance to enable selective application of phonological rules or other morphological logic. I enclose these markers in a pair of '`^`' symbols. These have a purely system-

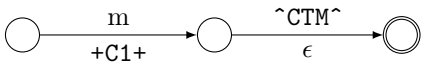
internal purpose—flagging morphemes for participation in rules—and thus are all removed as the final step in creating the XSMA FST. Some of the markers used are as follows:

- `^CTM^`: placed after an 'm' which interacts with the following vowel, e.g., as in *m-ema* → *mwema* '1.good'.
- `^PAST^`: used to mark the presence of the past morpheme *-li-* so that it can be changed to *-ku-* in the context of a preceding `^NEG^`, as in *si-li-sema* → *si-ku-sema* 'AN.1SG-NEG-PAST-say'.

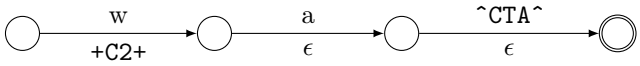
4.3 Class Prefixes

Since much of Swahili's morphology is driven by patterns of noun class agreement, establishing a uniform way of referring to the class prefixes is essential. XSMA makes the same basic distinction between nominal and verbal class prefixes as the one described in section 2.1. Since these prefixes are used throughout the system, they are stored in XFST definitions like (27) - (29):

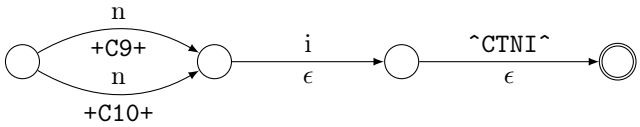
(27)
```
define CN1 [ [m "^CTM^"]:["+C1+"] ] ;
```



(28)
```
define CN2 [ [{wa} "^CTA^"]:["+C2+"] ] ;
```



(29)
```
define CN9 [ [{ni} "^CTNI^"]:["+C9+"] ] ;
define CN10 [ [{ni} "^CTNI^"]:["+C10+"] ] ;
```



The pattern in (27) - (29) should be clear: on the upper side we define the prefix (along with hints to phonological rules) and relate it the lower side, which consists of the information contributed by the morpheme (e.g., class information). In (29) we can see one of the advantages of using a finite-state approach to morphological analysis: the similarity in the definitions `CN9` and `CN10` is exploited in order to make a compact network representation of both morphemes. All of these prefix definitions are combined into one disjunction called `CNP`, which is used in other parts of the system that require the nominal class prefixes:

(30)
```
define CNP [
    CN1  |
    CN2  |
      ⋮
    CN17 |
    CN18
] ;
```

This `CNP` definition encodes a new FST: one built up from all the smaller FSTs defined for each class prefix. The verbal class prefixes are defined similarly, as shown in the following examples (cf. Table 3 on page 9).

(31)
```
define CV5 [ [{li} "^CTIV^"]:["+C5+"] ] ;
```

(32)
```
define CV6 [ [{ya} "^CTAV^"]:["+C6+"] ] ;
```

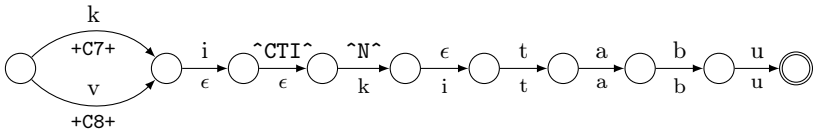
As with the nominal class prefixes, all of these verbal class prefixes are together defined as `CVP` so that they can be referenced jointly.

4.4 Nouns and Adjectives

Nouns and adjectives consist of a class prefix and a stem. As described in section 2.2, the prefix is basically fixed for nouns, whereas adjective stems can be combined with any of the

class prefixes. In the regular expressions for nouns, then, explicit reference is made to the class-pair to which the noun belongs, as in (33):

(33) `[ [ CN7 | CN8 ] "^N^":{ki} {tabu} ]`



The only segment of this expression which needs discussion is `"^N^":{ki}`. The `^N^` marker is placed at the beginning of every noun, in order to distinguish noun stems from adjective stems and is ultimately removed from the network in the final clean-up step. On the analysis side, `{ki}` is present simply because the 'dictionary' form of the word under consideration is *kitabu*, not *-tabu*. , as below:[12]

Adjectives are somewhat simpler to represent, since they take prefixes for any class, and their analysis form is simply the stem. There are, however, two classes of adjectives: those that use the nominal class prefixes for agreement, and those that use the verbal class prefixes. This distinction must be made in the regular expressions for each adjective, as in (34) and (35):

(34) `[ CNP "^NADJ^":0 {dogo} ]`

(35) `[ CVP "^VADJ^":0 {ote} ]`

The adjective *-dogo* 'small' takes the nominal class prefixes, as denoted by the use of the XFST definition `CNP`. *-ote* 'all' is an example of the special class of adjectives which take the verbal class prefixes, hence the presence of `CVP`.

[12]I could, of course, have taken *kitabu* to be unanalyzable, contributing class information without any mention of prefixes in singular and plural entries. Even with this strategy, however, a way would need to be found to accept the plural form *vitabu* while delivering the singular form *kitabu* in the analysis string.

Nouns and adjectives are eventually unioned into `Nouns` and `Adjs` definitions, respectively, using the strategy in (36):[13]

(36)
```
define Adjs [
   [ CVP "^VADJ^":0 {a} ]    |  # a ‘of‘
   [ CVP "^VADJ^":0 {ake} ]  |  # ake ‘POSS.3P.Sg‘
   [ CVP "^VADJ^":0 {ako} ]  |  # ako ‘POSS.2P.Sg‘
                ⋮
   [ CNP "^NADJ^":0 {zuri} ]    # zuri ‘fine‘
] ;
```

4.5 Verb Stems and Verb Templates

Swahili verbs exhibit an incredible amount of morphological variation, and XSMA handles only a subset of all possible forms. The verbal system is separated into basically two sections: (a) the set of stems, and (b) templates which incorporate the stems into a large number of morphologically-valid forms. Defining the stems is mostly straightforward, keeping in mind that there are two classes of verbs in Swahili (Bantu- and Arabic-stem). This situation does affect how morphemes are added to the stem (cf. section 2.3.6); thus different groups must be defined. Two such groups are given below:

(37)
```
define VStemBantu [
   {anguk} 0:a |  # anguka ‘fall‘
   {fik} 0:a   |  # fika ‘arrive‘
          ⋮
   {tumi} 0:a  |  # tumia ‘use‘
   {waz} 0:a      # waza ‘ponder‘
] ;
```

(38)
```
define VStemArabicE [
   {sameh} 0:e    # samehe ‘forgive‘
] ;
```

[13] In the XFST language, arbitrary comments may be added to the source code following the ‘#’ symbol. I have included some of these merely in order to gloss words used in the examples.

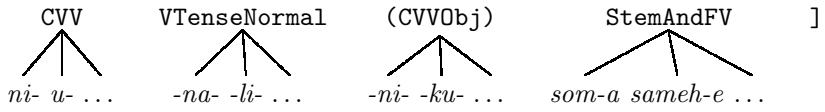
Various other definitions must also be built up before we can construct the verb templates. For example, the special subject and object markers used for agreement with animate nouns (cf. Table 4) need to be defined as in (39) and readied for insertion into a template along with the other verbal class prefixes.[14]

(39)
```
define CV1Sg     [ {ni}:["+Anim+" "+Sg+" "+P1+"] ] ;
define CV2SgSubj [ u:["+Anim+" "+Sg+" "+P2+"] ] ;
define CV2SgObj  [ {ku}:["+Anim+" "+Sg+" "+P2+"] ] ;
          ⋮
define CV2PlObj  [ {wa}:["+Anim+" "+Pl+" "+P2+"] ] ;
define CV3Pl     [ {wa}:["+Anim+" "+Pl+" "+P3+"] ] ;
```

With these and other morpheme categories defined appropriately, it is possible to define a set of regular expressions which act as templates, pulling in the morphemes which can legitimately occupy various slots in a given construction. The template for one such construction is given in (40):

(40) `[ CVV VTenseNormal (CVVObj) StemAndFV ]`

(40) defines a basic declarative verb with optional object marker. Given the number of class prefixes defined in `CVV` and `CVVObj` and the number of tenses defined in `VTenseNormal`, this simple template alone will construct an FST which can analyze thousands of possible forms for each stem in the lexicon. (41) depicts this exponentiality hidden in the template:

(41)


The full set of templates like (40) are ultimately combined into one `Verbs` definition, readying a huge class of forms for inclusion into the final `Lexicon` via disjunction with other

[14] All the verbal class prefixes are disjoined into two definitions, `CVV` for subject markers and `CVVObj` for object markers.

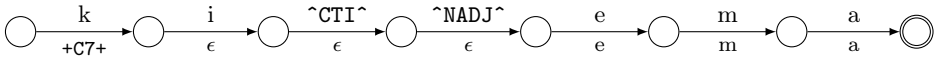
lexical components. The network must then be inverted to ready it for the application of replacement rules:[15]

(42)
```
read regex [ [Nouns | Adjs | Verbs ] 0:"+End+" ] ;
invert net
define Lexicon
```

4.6 Phonological and Morphological Rules

The network as defined by `Lexicon` in (42) is not, as it stands, complete. The analysis side of the network (i.e., lower side of the network before the inversion) is already fully specified, but not all strings on the opposite side (the lexical or upper side) are valid Swahili words.

At this stage, if we were to query the network for the Swahili word corresponding to the analysis `+C7+ema`, we would not get the expected `chema` ('7.good') but rather the very odd `ki^CTI^^NADJ^ema`, which corresponds to the analysis `+C7+ema` (in the FST we built up from the `NCP` and `Adjs` definitions):

(43)


The system, of course, does not know that the prefix *ki-* should become *ch-* before vowels. Fortunately, XFST's conditional replacement operator is well-suited to ensuring that paths in the network like `ki^CTI^^NADJ^ema` get replaced with the correct surface path `chema`, as in (44):

(44)
```
[ [k i] -> [c h] || _ "^CTI^" "^NADJ^" Vowel ]
```

(44) says that `ki` should be replaced with `ch` when followed by (a) the marker `^CTI^`, (b) the marker `^NADJ^`, and (c) a vowel. The marker `^CTI^` plays an important role, since the

[15] In (42), only three lexical categories are shown in order to save space; XSMA can also handle other word classes.

phonological rule described does not apply in just any *ki*-environment, but rather is restricted to environments featuring the *class 7 prefix ki-* in particular.

Figure 4 (p. 31) shows how the FST in (43) is transformed via composition with (44) into a network which contains the correct surface string. Figure 4 also contains a rule which deletes the markers `^CTI^` and `^NADJ^` from the network entirely. (Once the rules have been applied, such markers play no role, and serve only to clutter the lexical side of the transducer with linguistically-meaningless symbols.)

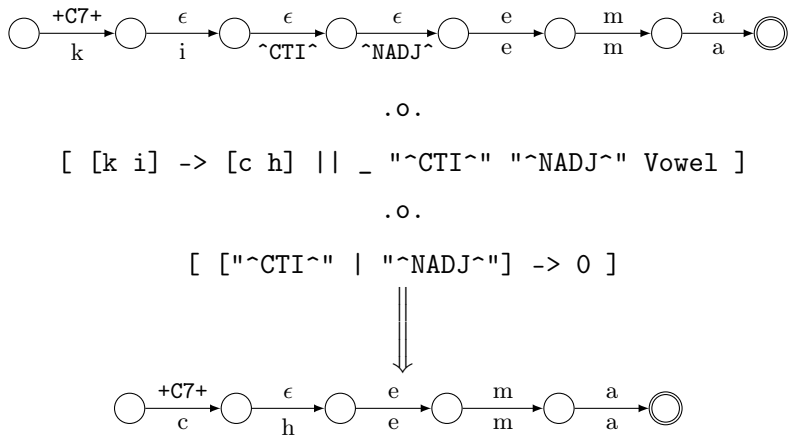


Figure 4: The effect of a phonological rule on an FST fragment

Numerous other phonological rules must be implemented in order to make sure the lexical side of the XSMA FST contains valid surface forms of Swahili words. The full set of rules can be found in the source code (Appendix A); for the moment I will simply give the XFST replacement rules corresponding to the phonological rules listed for the class-pair 9/10 prefixes in section 2.4:[16]

[16] In these rules, a number of definitions are used whose meaning and extension should be obvious, though a few are not: `CTNI` contains the class-pair 9/10 environment marker, and `NA` references markers denoting either the beginning of a noun stem or a nominal-prefix-type adjective stem.

(45)
```
[ i -> 0 || n _ CTNI NA Consonant ] .o.
[ i -> y || n _ CTNI NA Vowel ] .o.
[ y -> j || n _ CTNI NA e ] .o.
[ n -> 0 || _ CTNI NA VoicelessOnset Nucleus Syll+ ] .o.
[ n -> m || _ CTNI NA Labial ] .o.
[ w -> b || CTNI NA _ ] .o.
[ [l | r] -> d || CTNI NA _ ]
```

Through the application of these and many other rules (including, as mentioned above, the replacements which remove all helper markers, e.g., `^NADJ^`), the full `Rules` definition can be built, and applied to the `Lexicon` as follows:

(46)
```
read regex [
   Lexicon .o.
   Rules
] ;
```

The resulting network constitutes the full XSMA system, ready for analysis or generation.

5 Issues Uncovered in Implementation

In this section, I will reflect on issues and challenges stemming from the approach I have laid out above. While some of these issues are explicated with reference to idiosyncracies in Swahili, none are ultimately cross-linguistically unique. I hope, therefore, that these comments are relevant to computational morphology in general.

5.1 Representation of Linguistic Entities

The question of representation in linguistics is obviously too general to consider here. The situation in FST-based morphological analysis is thankfully somewhat constrained to begin with, given that regular expressions and XFST rules are essentially strings of characters. However, it is still appropriate to ask in what sense a sequence of linguist-defined tags represents the morphological structure of a Swahili word, or in what sense a conditional replacement rule (as exemplified in (45)) represents the phonological rules familiar from traditional linguistic descriptions.

In section 3.2, I said that the goal of morphological analysis was to perform the operation depicted in (22):

(22) `nimezipika` ⟶ `+Anim+ +Sg+ +P1+ +Perf+ +C10+ pika`

The result of the operation is a set of abstract tags intended to describe the morphological contributions of morphemes like *ni-* or *-me-*. Clearly, the tags themselves are arbitrary, and it is not important to argue for their particular form.

More interestingly, one morpheme (e.g. *ni-*) can generate multiple morphological tags, in this case `+Anim+`, `+Sg+`, and `+P1+`. This mapping reflects a fundamental design decision of the system; in XSMA, emphasis is placed on de-synthesizing whatever functional information is contributed by a morpheme, such that a minimal set of linguistically-sensible tags is required. I define two number tags and three person tags, rather than the six tags which would be

required to define the synthetic version. Thus the analysis string produced by XSMA represents neither morphemes-plus-stem, nor even the *meaning* of the morphemes plus stem, but rather the *functional information* contributed by each morpheme, in morphemic order. Is this the most desirable kind of morphological analysis? It at least has the benefit of elucidating the possibly-multiple contributions of morphemes; for this reason, the analyses provided by XSMA are natural inputs to frameworks based on functional theories of syntax like LFG.

Phonological rules raise other issues of representation. The replacement rules given in the previous section are superficially similar to the linguists' phonological rules, but there is a difference: phonological rules are supposed to operate on *phonemes*, but the rules of XSMA manipulate characters of standard Swahili orthography. XSMA does in some sense *model* the phonological rules, but this is only possible due to the relatively clear relationship, in Swahili, between orthography and phonology. It is therefore important to be clear that the rules of XSMA are really *spelling* rules, but these happen to describe the phonological facts quite well. In another language, the task of modelling morpho-phonological behavior in ASCII strings may be much more complicated, and lead to a set of rules which are less transparent.

5.2 Non-concatenative morphology

Several morphological operations in Swahili are considered non-concatenative; that is, they do not simply glue morphemes together. Reduplication is a classic example of non-concatenative morphology, wherein a certain element of fixed or arbitrary length is repeated. The requirement that an arbitrary element be repeated poses a challenge for FST-based analyzers, since at any given state in the transducer, the system is entirely ignorant of which arcs were traversed to reach that state.

Novotna (2000) describes a number of cases of reduplication in Swahili (e.g., verbal base reduplication), along with their semantic contribution. One pattern which emerges clearly is that reduplication is a highly variable and context-sensitive operation in Swahili, and in some cases the meaning of a reduplicated form cannot be precisely predicted. For these reasons,

reduplication was not treated formally within XSMA, with the result that reduplicated verbal bases must simply be listed alongside their simplex versions.

However, reduplication is a phenomenon which *can* in fact be implemented in an FST-based system without resorting to listing reduplicated items. Walther (2000) gives a method for extending FSNs to cope with reduplication, and his method can also be applied to FSTs. XFST, in fact, has its own solution built into the system via the `compile-replace` command (to which Beesley and Karttunen (2000:375) devote a whole chapter), which facilitates the inclusion of sub-FSTs used, e.g., to loop through arcs of the main FST, thereby producing reduplicated elements.

XSMA makes use of this technique in order to analyze another instance of non-concatenative morphology in Swahili: the demonstratives. The demonstrative with a distal semantics in Swahili is *-le*, to which the verbal class markers can be prefixed straightforwardly. The demonstrative with a proximal semantics, however, exhibits non-concatenative behavior, as exhibited in (47) and (48):

(47) *mti* *h-uu*
3.tree DEM.PROX-3
'this tree'

(48) *maembe* *h-aya*
6.mango DEM.PROX-6
'these mangoes'

The word *h-* combines with the verbal class prefix, but also epenthesizes the vowel of that verbal class prefix, obeying a sort of vowel harmony. To model these facts, a regular expression is first defined which describes the non-concatenative behavior:

(49) `[h "^HXX^" "^[" "["]:0 CVH ["]" {^2} "^]"]:{hxxx}`

(49) defines, on the upper side, strings like `h^HXX^^[[ya]^2^]`, containing special delimiters which designate a section of the FST for `compile-replace` to operate on. Everything

within these delimiters is treated as a regular expression in its own right; thus, since `^`x is a regular expression operation which repeats whatever is in its scope x times, `compile-replace` is able to take the verbal class prefix `ya` and turn it into `yaya`.

The strategy used here is essentially to reduplicate the class prefix (`CVH`), then to use a conditional replacement rule to delete any consonant immediately following `^HXX^`. In other words, `compile-replace` first creates a network which contains forms like `h^HXX^yuyu`, thereby feeding another rule which deletes `^HXX^` + Consonant. The final FST then contains the appropriate surface forms like `huyu`, `haya`, etc...

While this treatment of *h-* is not particularly useful in terms of the amount of work saved by using the technique, it does prove that non-concatenative morphology can in general be handled in this approach, using `compile-replace`.

5.3 Derivational morphology

Another aspect of Swahili which poses a challenge to any computational analysis is the semi-regular nature of its derivational morphology. Verb stems, for example, may be augmented in a variety of ways. Passivization is regularly signalled by the morpheme *-w-*, epenthesized before the final vowel, and could thus be thought of as a derivational operation in Swahili. (50) shows a passive form:

(50) *Kitabu* *ki-na-som-w-a*
7.book 7.SBJ-PRES-read-PASS-FV
'The book is being read/studied'

Stems may also be augmented with *-sh-* to show a causative meaning, or with *-k-* to exhibit a stative one, as in (51) and (52):

(51) *Ni-na-wa-som-esh-a* *wanafunzi*
AN.1SG.SBJ-PRES-AN.3PL.OBJ-study-CAUSE-FV 2.student
'I am instructing the students.'

(52) *Mwandiko* *u-na-som-ek-a*
3.handwriting 3.SBJ-PRES-read-STAT-FV
'The handwriting is readable/legible.'

The semantic contribution of these morphemes is not entirely transparent: *-somesha* means 'instruct X' more intuitively than it means 'cause X to read'; likewise, *-someka* is not describing a state of being read, but something rather more like its ability to be read.

While the morphological rules which conjoin these semantic-shift morphemes are regular, the meanings produced are not always so. Thus, I have elected to forego the inclusion of this kind of morphology in XSMA itself, with the exception of the more regular passive..

5.4 Word-internal constraints

The verb templates described in section 4.5 do a lot of work to constrain the types of forms which are accepted and generated by XSMA. However, there is still a certain amount of over-generation inherent in the system, due to the interaction possible between different elements in the verbal templates. In the template below, for example, it is possible for the verb to exhibit both a subject and an object marker:

(40) `[ CVV VTenseNormal (CVVObj) StemAndFV ]`

In general, we want `CVV` and `CVVObj` to vary freely with respect to one another, to allow for the different combinatoric possibilities used to express a variety of situations. There is one important exception: the same morpheme is not allowed to appear in both positions, as (53) illustrates. In expressions whose intended meaning is that of (53), the reflexive marker *-ji-* must be used, as in (54).

(53) **Ni-na-ni-on-a*
AN.1SG.SBJ-PRES-AN.1SG.OBJ-see-FV
'*I see me'

(54) ***Ni****-na-****ji****-on-a*
AN.1SG.SBJ-PRES-REFL.OBJ-see-FV
'I see myself'

These facts are difficult to model using a template like the one in (40). The use of definitions like `CVV` and `CVVObj` give us a powerful way to define an entire class of verb forms in one line, but they do not allow us to state concisely the generalization about the reflexive examples, which is essentially a constraint on the co-occurrence of verbal morphemes.

Reflexives are not the only constructions which exhibit this kind of word-internal constraint; general relatives, for example, pose a similar challenge for the system. Such constructions can easily be rejected as ungrammatical on functional grounds, however, by some other system (e.g., LFG's XLE framework), and XSMA itself does not take a stand on the morphological validity of forms like **ni-na-ni-on-a*.

6 Conclusion

In this paper, I have given a sketch of Swahili morphology and shown how a large portion of its features can be adequately modeled using finite-state transducers. None of this is completely novel; FSTs are already popular tools in describing morphological systems, and the XFST framework was developed with just such applications in mind. Swahili morphology in particular has already been given the FST treatment in a system called SWATWOL (Hurskainen 1992), using an older finite-state compiler (Dalrymple et al. 1987).

The advantages of the XSMA system have primarily to do with how easily it can be integrated with other frameworks. Given that it is developed as an XFST transducer, it is immediately available for use in XLE grammars. Furthermore, the technique of storing lexical information in a standard linguist's database like Toolbox means that the lexicon can be maintained without any understanding of the XSMA code itself.

One of the fundamental disadvantages of FST-based approaches in general is their reliance on a hand-compiled lexicon. To this point, work is currently in progress to automatically parse words from an online Swahili dictionary and store them in the appropriate format; XSMA will subsequently be tested on Swahili corpora in order to gather statistics about its performance and uncover issues which have not surfaced thus far in development. A completely different approach is being taken by the Linguistica project (Goldsmith (2000)), which aims at using unsupervised machine learning techniques to induce morphological information directly from corpora.

Swahili is a language well-suited for the kind of morphological exploration undertaken in this paper. It is a morphologically-heavy language, but for the most part its operations can be defined with a small number of rules and templates. FSTs and XFST in particular provide a fruitful basis for continued research in the area of computational morphological analysis, and I hope in this paper to have pointed out a number of the conceptual and practical issues involved in building a morphological analyzer for a language like Swahili.

References

M. Aronoff, I. Meir, C. Padden, and W. Sandler. Classifier constructions and morphology in two sign languages. *Perspectives on Classifier Constructions in Sign Languages*, pages 53–84, 2003.

E. Ashton. *Swahili Grammar (Including Intonation)*. London: Longman, 1987.

K.R. Beesley and L. Karttunen. Finite-state non-concatenative morphotactics. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 191–198. Association for Computational Linguistics, 2000.

K.R. Beesley and L. Karttunen. *Finite State Morphology*. CSLI, 2003.

B. Bickel, B. Comrie, and M. Haspelmath. The Leipzig glossing rules. Conventions for interlinear morpheme by morpheme glosses. *Max Planck Institute for Evolutionary Anthropology and Department of Linguistics, University of Leipzig*, 2008.

J. Bresnan and S.A. Mchombo. The lexical integrity principle: Evidence from Bantu. *Natural Language & Linguistic Theory*, 13(2):181–254, 1995.

D. Crouch, M. Dalrymple, R. Kaplan, T. King, J. Maxwell, and P. Newman. XLE documentation. 2007. URL `http://www2.parc.com/isl/groups/nltt/xle/doc/xle_toc.html`.

M. Dalrymple, L. Karttunen, and S. Shaio. DKIMMO: A morphological analyzer using two-level rules. *R. Kaplan and L. Karttunen: Computational Morphology, Course Script LI283*, 1987.

G. De Pauw and G.M. De Schryver. Improving the computational morphological analysis of a Swahili corpus for lexicographic purposes. *Lexikos*, 18(0), 2009.

K. Demuth. Bantu noun class systems: Loanword and acquisition evidence of semantic productivity. *Systems of Nominal Classification*, pages 270–292, 2000.

P. Edelsten, L. Marten, and N. Kula. Swahili relative clauses. 2010. URL `http://www.essex.ac.uk/linguistics/department/events/CWH/handouts/Edelsten-Kula-Lutz.pdf`.

J. Goldsmith. Linguistica: An automatic morphological analyzer. In *Proceedings of 36th Meeting of the Chicago Linguistic Society*. U. Chicago, 2000.

J. Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198, 2001.

C. Gussenhoven and H. Jacobs. *Understanding Phonology*. London: Arnold Hodder, 2005.

A. Hurskainen. A two-level computer formalism for the analysis of Bantu morphology. *Nordic Journal of African Studies*, 1(1):87–119, 1992.

A. Hurskainen. Disambiguation of morphological analysis in Bantu languages. In *Proceedings of the 16th Conference on Computational linguistics*, volume 1, pages 568–573. Association for Computational Linguistics, 1996.

A. Hurskainen. Optimizing disambiguation in Swahili. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 254. Association for Computational Linguistics, 2004.

A. Hurskainen. Solutions for handling non-concatenative processes in bantu languages. *Inquiries into Words, Constraints and Contexts*, page 45, 2005.

M.A. Mohammed. *Modern Swahili Grammar*. East African Education Press, 2001.

F. Mpiranya. *Swahili Phonology Reconsidered in a Diachronical Perspective: the Impact of Stress on Morphonemics and Syllable Structure*. R. Köppe, 1995.

J. Novotna. Reduplication in Swahili. In *Swahili Forum VII. (Afrikanistische Arbeitspapiere)*, volume 64, pages 57–73. Köln: Institut für Afrikanistik, Universität Köln, 2000.

E.C. Polomé. Swahili language handbook. 1967.

S. Robinson, G. Aumann, S. Bird, et al. Managing fieldwork data with Toolbox and the Natural Language Toolkit. *Language Documentation and Conservation*, 1(1):44–57, 2007.

J. Russell and D.V. Perrott. *Teach Yourself Swahili*. NTC Pub. Group, 1996.

T.C. Schadeberg. *A Sketch of Swahili Morphology*. Foris Publications, 1984.

T.C. Schadeberg. Number in Swahili grammar. In *Swahili Forum VIII (Afrikanistische Arbeitspapiere)*, volume 68, pages 7–16. Köln: Institut für Afrikanistik, Universität Köln, 2001.

B. Wald. Swahili and the Bantu languages. In B. Comrie, editor, *The World's Major Languages*, pages 991–1014. Oxford University Press, 1987.

M. Walther. Finite-state reduplication in one-level prosodic morphology. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 296–302. Morgan Kaufmann Publishers Inc., 2000.

Appendix A: Code Listings

Most of the files which make up the XSMA codebase are listed in this appendix.[17] Certain files, such as scripts which pull data from the Toolbox database, are not included. It should be pointed out that the code included is in a state of active development, and is not stable. Given this situation, the lexicon has been kept extremely small in order to facilitate testing.

A.1 build-morphology.fsts

```
### BEGINNING BUILD OF XSMA ###

clear stack
set flag-is-epsilon ON

### LOADING DEFINITIONS ###
source rules/definitions.fsts

### INITIALIZING SWAHILI LEXICON ###
source lex/sublex-class.fsts
source lex/auto-nouns.fsts
source lex/sublex-nouns.fsts
source lex/auto-adjs.fsts
source lex/auto-verbs.fsts
source lex/sublex-verbs.fsts
source lex/auto-advs.fsts
source lex/auto-preps.fsts
source lex/auto-etc.fsts
source lex/misc.fsts

### MAKING NON-CONCATENATIVE FORMS ###
source lex/nonconcat.fsts
invert net
define NonConcatLex
source rules/nonconcat-pre.fsts
define NonConcatPreClean
read regex [ NonConcatLex .o. NonConcatPreClean ];
compile-replace lower
define NonConcatPost
source rules/nonconcat-post.fsts
read regex [NonConcatPost .o. NonConcatRules ];
invert net
define NonConcat

### BUILDING SWAHILI LEXICON ###
read regex [ [Nouns | Adjs | Verbs | Advs | Preps | Etc | Misc | NonConcat]
             0:"+End+" ] ;
invert net
define Lexicon

### APPLYING SWAHILI RULES ###

source rules/phonology.fsts
source rules/verbal-morph.fsts
source rules/irregulars.fsts
source rules/cleanup.fsts

```

[17] This code is included for the purpose of academic research, and permission must be sought before using any portion of it in any implementation whatsoever.

```
48 define Rules [ Phonology .o. VerbalMorph .o. Irregulars .o. Cleanup ] ;
49
50 read regex [
51     Lexicon .o.
52     Rules
53 ] ;
54
55 ### XSMA READY ###
56
57 save stack xsma.fst
```

A.2 rules/definitions.fsts

```
1  # +xxx+ are morphological tags used in XLE
2  # ^xxx^ are used by morphological rules only, and are described in rules
3  # definitions
4
5  define CTI [
6      "^CTI^" |    # put after an 'i' that should contract with following vowel
7      "^CTI5^"     # put after a class 5 'i' that should contract with following
8                   # vowel
9  ];
10 define NA [
11     "^N^"    |   # put before a noun stem
12     "^NADJ^"     # put before an adjective stem which takes noun agreement
13 ];
14 define CTSymbol [
15     CTI          | # defined above
16     "^CTM^"      | # put after an 'm' that should contract with following vowel
17     "^CTA^"      | # put after a class 1 'a' that should contract with a
18                    # following vowel
19     "^CTU^"      | # put after a class 11 'u' that should contract with a
20                    # following vowel
21     "^CTNI^"     | # put after a class 9/10 'ni' that should change in various
22                    # ways in e.g. 9/10
23     "^CTIV^"     | # put after an 'i' in verb prefix
24     "^CTAV^"     | # put after an 'a' in verb prefix
25     "^CTUV^"     | # put after an 'u' in verb prefix
26     "^CTMV^"     | # put after an 'm' in verb prefix
27     "^CTBL^"       # blocks contraction e.g. as in kiongozi !-> chongozi
28 ] ;
29 define Symbol [
30     "^["         |   # non-concat 1
31     "^]"         |   # non-concat 2
32     CTSymbol     |   # defined above
33     NA           |   # defined above
34     "^PN^"       |   # used before Proper name for class stuff
35     "^VADJ^"     |   # marker before adj stem which takes verb agreement
36     "^NEG^"      |   # after negative to combine with ^PAST^ to make ku
37     "^PAST^"     |   # before the 'li' past tense
38     "^PERF^"     |   # before the 'me' perfect tense
39     "^PASS^"     |   # before the 'w' passive morpheme
40     "^ASTEM^"    |   # used to mark end of arabic stem; needed for passivization
41     "^MONOV^"    |   # before the single consonant of a monosyllabic verb stem,
42                     # to turn to 'ku'
43     "^MONOVOK^" |   # before the single consonant of a monosyllabic verb stem,
44                     # to keep from turning to 'ku'
45     "^OREF^"     |   # before the -o of reference (used in relatives, h--o...)
46     "^HXX^"          # used for hii, huu, haya setup
47 ] ;
48
```

```
# Other misc definitions
define Letter [a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|u|v|w|x|y|z] ;
define Vowel [a|e|i|o|u] ;
define Consonant [Letter & ~Vowel] ;

# Phonological definitions
define HighV [i|u] ;
define LowV [a|o] ;
define MidV [e] ;
define FrontV [i|e] ;
define BackV [a|o|u] ;
define Labial [b|f|m|p|v|w];
define VoicedCons [b|d|g|j|l|m|n|r|v|w|z];
define VoicelessCons [Consonant & ~VoicedCons];

# Metrical definitions
define SyllNas [m | n];
define LRW [l|r|w];
define VoicedOnset [[b (LRW)]|[d (h|r|w)]| [g (h|LRW)]|j|l|m|[n (y)]|r|[v (y)]|w|y|z];
define VoicelessOnset [[c (h)]|[f (LRW|y)]|h|[k (LRW|y)]|[p (LRW|y)]|
    [s (h|k|l|m|n|p|w)]|[t (LRW)]];
define Onset [VoicedOnset | VoicelessOnset];
define Rhyme [ Vowel ];
define Coda [ Consonant & ~[c|y|x|j|h|w|q|v] ];
define Nucleus [Rhyme (Coda)];
define Syll [ [(Onset) Nucleus] | SyllNas ];
define SyllCons [Onset Nucleus];
```

A.3 lex/sublex-class.fsts

```
# NOMINAL CLASS PREFIXES

define CN1 [ [m "^CTM^"]:["+C1+"] ] ;
define CN2 [ [{wa} "^CTA^"]:["+C2+"] ] ;

define CN3 [ [m "^CTM^"]:["+C3+"] ] ;
define CN4 [ [{mi} "^CTI^"]:["+C4+"] ] ;

define CN5 [ [{ji} "^CTI5^" ]:["+C5+"] ] ;
define CN5Eps [ 0:["+C5+"] ] ;
define CN6 [ [{ma} "^CTA^"]:["+C6+"] ] ;
define CN6Eps [ {ma}:["+C6+"] ] ; # no contraction for epsilon nouns

define CN7 [ [{ki} "^CTI^"]:["+C7+"] ] ;
define CN8 [ [{vi} "^CTI^"]:["+C8+"] ] ;

define CN9 [ [n i "^CTNI^"]:["+C9+"] ] ;
define CN10 [ [n i "^CTNI^"]:["+C10+"] ] ;
define CN9Eps [ 0:["+C9+"] ] ;
define CN10Eps [ 0:["+C10+"] ] ;

define CN11 [ [u "^CTU^"]:["+C11+"] ] ;
define CN11Adj [ [m "^CTM^"]:["+C11+"] ] ;
define CN12 [ [n i "^CTNI^"]:["+C12+"] ] ;
define CN12Eps [ 0:["+C12+"] ] ;

define CN13 [ [u "^CTU^"]:["+C13+"] ] ;
define CN13Adj [ [m "^CTM^"]:["+C13+"] ] ;
define CN14 [ [{ma} "^CTA^"]:["+C14+"] ] ;

define CN15 [ [{ku} "^CTU^"]:["+C15+"] ] ;
```

```
32
33 define CN16 [ [{pa} "^CTA^"]:["+C16+"] ] ;
34
35 define CN17 [ [{ku} "^CTU^"]:["+C17+"] ] ;
36
37 define CN18 [ [{mu} "^CTU^"]:["+C18+"] ] ;
38
39 # These are defined for adjective agreement purposes
40 define CNP [
41     CN1 |
42     CN2 |
43     CN3 |
44     CN4 |
45     CN5 |
46     CN6 |
47     CN7 |
48     CN8 |
49     CN9 |
50     CN10 |
51     CN9Eps |
52     CN10Eps |
53     CN11Adj |
54     CN12 |
55     CN12Eps |
56     CN13Adj |
57     CN14 |
58     CN15 |
59     CN16 |
60     CN17 |
61     CN18
62 ] ;
63
64 # VERBAL CLASS PREFIXES
65
66 define CV1M [ [w | [m "^CTMV^"]]:["+C1+"] ] ;
67 define CV1Yu [ [{yu} "^CTUV^"]:"+C1+" ];
68 define CV1 [CV1M | CV1Yu];
69 define CV2 [ [{wa} "^CTAV^"]:["+C2+"] ] ;
70
71 define CV3 [ [u "^CTUV^"]:["+C3+"] ] ;
72 define CV4 [ [i "^CTIV^"]:["+C4+"] ] ;
73
74 define CV5 [ [{li} "^CTIV^"]:["+C5+"] ] ;
75 define CV6 [ [{ya} "^CTAV^"]:["+C6+"] ] ;
76
77 define CV7 [ [{ki} "^CTIV^"]:["+C7+"] ] ;
78 define CV8 [ [{vi} "^CTIV^"]:["+C8+"] ] ;
79
80 define CV9 [ [i "^CTIV^"]:["+C9+"] ] ;
81 define CV10 [ [{zi} "^CTIV^"]:["+C10+"] ] ;
82
83 define CV11 [ [u "^CTUV^"]:["+C11+"] ] ;
84 define CV12 [ [{zi} "^CTIV^"]:["+C12+"] ] ;
85
86 define CV13 [ [u "^CTUV^"]:["+C13+"] ] ;
87 define CV14 [ [{ya} "^CTAV^"]:["+C14+"] ] ;
88
89 define CV15 [ [{ku} "^CTUV^"]:["+C15+"] ] ;
90
91 define CV16 [ [{pa} "^CTAV^"]:["+C16+"] ] ;
92
93 define CV17 [ [{ku} "^CTUV^"]:["+C17+"] ] ;
```

```
94
95  define CV18 [ [{mu} "^CTUV^"]:["+C18+"] ] ;
96
97  # These are defined for adjective w/ verb agreement purposes
98  define CVP [
99      CV1 |
100     CV2 |
101     CV3 |
102     CV4 |
103     CV5 |
104     CV6 |
105     CV7 |
106     CV8 |
107     CV9 |
108     CV10 |
109     CV11 |
110     CV12 |
111     CV13 |
112     CV14 |
113     CV15 |
114     CV16 |
115     CV17 |
116     CV18
117 ] ;
118
119 # For use with hxxx, hxxo
120 define CVH [
121     CVP - CV1M
122 ];
```

A.4 lex/auto-nouns.fsts

```
1   define AutoNouns [
2       [ [ CN9Eps | CN10Eps ] "^N^":0 {barua} ] |  # barua ‘letter‘
3       [ [ CN7 | CN8 ] "^N^":{ch} {akula} ] |  # chakula ‘food‘
4       [ [ CN9Eps | CN10Eps ] "^N^":0 {chapati} ] |  # chapati ‘chapati‘
5       [ [ CN7 | CN8 ] "^N^":{ch} {umba} ] |  # chumba ‘room‘
6       [ [ CN5Eps | CN6Eps ] "^N^":0 {embe} ] |  # embe ‘mango‘
7       [ [ CN9Eps | CN10Eps ] "^N^":0 {shule} ] |  # shule ‘school‘
8       [ [ CN9Eps | CN10Eps ] "^N^":0 {sanaa} ] |  # sanaa ‘art‘
9       [ [ CN5 | CN6 ] "^N^":0 {yai} ] |  # yai ‘egg‘
10      [ [ CN5 | CN6 ] "^N^":0 {jani} ] |  # jani ‘leaf‘
11      [ [ CN9Eps | CN10Eps ] "^N^":0 {jibini} ] |  # jibini ‘cheese‘
12      [ [ CN5 | CN6 ] "^N^":{j} {iko} ] |  # jiko ‘stove‘
13      [ [ CN5 | CN6 ] "^N^":{j} {ino} ] |  # jino ‘tooth‘
14      [ [ CN9Eps | CN10Eps ] "^N^":0 {jinsi} ] |  # jinsi ‘way‘
15      [ [ CN9Eps | CN10Eps ] "^N^":0 {meza} ] |  # meza ‘table‘
16      [ [ 0:"+C16+" ] "^N^":0 {pahali} ] |  # pahali ‘place‘
17      [ [ CN5 | CN6 ] "^N^":{ji} {we} ] |  # jiwe ‘stone‘
18      [ [ CN7 | CN8 ] "^N^":{ki} {kombe} ] |  # kikombe ‘cup‘
19      [ [ CN7 | CN8 ] 0:"+Anim+" "^N^":{ki} "^CTBL^":0 {ongozi} ] |  # kiongozi ‘leader‘
20      [ [ CN7 | CN8 ] "^N^":{ki} {pande} ] |  # kipande ‘piece‘
21      [ [ CN7 | CN8 ] "^N^":{ki} {tabu} ] |  # kitabu ‘book‘
22      [ [ CN7 | CN8 ] "^N^":{ki} {ti} ] |  # kiti ‘chair‘
23      [ [ CN7 | CN8 ] "^N^":{ki} {toweo} ] |  # kitoweo ‘stew‘
24      [ [ CN7 | CN8 ] "^N^":{ki} {tu} ] |  # kitu ‘thing‘
25      [ [ CN7 | CN8 ] "^N^":{ki} "^CTBL^":0 {ungo} ] |  # kiungo ‘spice‘
26      [ [ CN5 | CN6 ] 0:"+Anim+" "^N^":0 {kunguru} ] |  # kunguru ‘crow‘
27      [ [ CN9Eps | CN10Eps ] 0:"+Anim+" "^N^":0 {mama} ] |  # mama ‘mother‘
28      [ [ CN9 | CN10 ] 0:"+Anim+" "^N^":{m} {bweha} ] |  # mbweha ‘fox‘
29      [ [ CN3 | CN4 ] "^N^":{m} {domo} ] |  # mdomo ‘mouth‘
```

```
30      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{m} {fanyikazi} ] |  # mfanyikazi ‘worker‘
31      [ [ CN3 | CN4 ] "^N^":{m} {ji} ] |  # mji ‘town‘
32      [ [ CN3 | CN4 ] "^N^":{m} {kate} ] |  # mkate ‘loaf‘
33      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{m} {ke} ] |  # mke ‘wife‘
34      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{m} {kulima} ] |  # mkulima ‘farmer‘
35      [ [ CN3 | CN4 ] "^N^":{m} {lango} ] |  # mlango ‘door‘
36      [ [ CN3 | CN4 ] "^N^":{m} {oshi} ] |  # moshi ‘smoke‘
37      [ [ CN3 | CN4 ] "^N^":{m} {oto} ] |  # moto ‘fire‘
38      [ [ CN3 | CN4 ] "^N^":{m} {oyo} ] |  # moyo ‘heart‘
39      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{m} {pishi} ] |  # mpishi ‘cook‘
40      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{m} {sichana} ] |  # msichana ‘girl‘
41      [ [ CN3 | CN4 ] "^N^":{m} {situ} ] |  # msitu ‘woodland‘
42      [ [ CN3 | CN4 ] "^N^":{m} {ti} ] |  # mti ‘tree‘
43      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{m} {toto} ] |  # mtoto ‘child‘
44      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{m} {tu} ] |  # mtu ‘person‘
45      [ [ CN3 | CN4 ] "^N^":{mu} {da} ] |  # muda ‘time‘
46      [ [ CN3 | CN4 ] "^N^":{mu} {hogo} ] |  # muhogo ‘cassava‘
47      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{m} {ume} ] |  # mume ‘husband‘
48      [ [ CN3 | CN4 ] "^N^":{mu} {wa} ] |  # muwa ‘sugarcane‘
49      [ [ CN9Eps | CN10Eps ] "^N^":0 {mvua} ] |  # mvua ‘rain‘
50      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{m} {vulana} ] |  # mvulana ‘boy‘
51      [ [ CN3 | CN4 ] "^N^":{mw} {aka} ] |  # mwaka ‘year‘
52      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{mw} {alimu} ] |  # mwalimu ‘teacher‘
53      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{m} {geni} ] |  # mgeni ‘guest‘
54      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{mw} {uzaji} ] |  # mwuzaji ‘seller‘
55      [ CN1 0:"+Anim+" "^N^":0 {anamke}:{mwanamke} ] |
56      [ CN2 0:"+Anim+" "^N^":0 {anawake}:{mwanamke} ] |  # mwanamke ‘woman‘
57      [ CN1 0:"+Anim+" "^N^":0 {anamume}:{mwanamume} ] |
58      [ CN2 0:"+Anim+" "^N^":0 {anawaume}:{mwanamume} ] |  # mwanamume ‘man‘
59      [ [ CN3 | CN4 ] "^N^":{mw} {ezi} ] |  # mwezi ‘month‘
60      [ [ CN9 | CN10 ] 0:"+Anim+" "^N^":{n} {dege} ] |  # ndege ‘bird‘
61      [ [ CN9 | CN10 ] "^N^":{n} {dizi} ] |  # ndizi ‘banana‘
62      [ [ CN9 | CN10 ] "^N^":{n} {goma} ] |  # ngoma ‘drum‘
63      [ [ CN9Eps | CN10Eps ] 0:"+Anim+" "^N^":0 {ng'ombe} ] |  # ng'ombe ‘cow‘
64      [ [ CN9 | CN10 ] "^N^":{n} {jaa} ] |  # njaa ‘hunger‘
65      [ [ CN9 | CN10 ] "^N^":{n} {jia} ] |  # njia ‘way‘
66      [ [ CN5 | CN6 ] "^N^":0 {nyoya} ] |  # nyoya ‘feather‘
67      [ [ CN9Eps | CN10Eps ] "^N^":0 {pesa} ] |  # pesa ‘money‘
68      [ [ CN9Eps | CN10Eps ] "^N^":0 {sauti} ] |  # sauti ‘voice‘
69      [ [ CN5 | CN6 ] "^N^":0 {shaka} ] |  # shaka ‘doubt‘
70      [ [ CN5 | CN6 ] "^N^":0 {shamba} ] |  # shamba ‘farm‘
71      [ [ CN9Eps | CN10Eps ] "^N^":0 {sifa} ] |  # sifa ‘flattery‘
72      [ [ CN5 | CN6 ] "^N^":0 {siku} ] |  # siku ‘day‘
73      [ [ CN9Eps | CN10Eps ] 0:"+Anim+" "^N^":0 {sungura} ] |  # sungura ‘rabbit‘
74      [ [ CN11 | CN12 ] "^N^":{u} {bao} ] |  # ubao ‘board‘
75      [ [ CN11 | CN12Eps ] "^N^":{u} {kuta} ] |  # ukuta ‘wall‘
76      [ [ CN11 | CN12 ] "^N^":{u} {devu} ] |  # udevu ‘beard‘
77      [ [ CN11 | CN12Eps ] "^N^":{u} {funguo} ] |  # ufunguo ‘key‘
78      [ [ CN13 | CN14 ] "^N^":{u} {gonjwa} ] |  # ugonjwa ‘disease‘
79      [ [ CN11 | CN12 ] "^N^":{u} {limi} ] |  # ulimi ‘tongue‘
80      [ [ CN13 | CN14 ] "^N^":{u} {ovu} ] |  # uovu ‘evil‘
81      [ [ CN11 | CN12Eps ] "^N^":{u} {pepo} ] |  # upepo ‘wind‘
82      [ [ CN11 | CN12 ] "^N^":{w} {avu} ] |  # wavu ‘net‘
83      [ [ CN11 | CN12 ] "^N^":{w} {imbo} ] |  # wimbo ‘song‘
84      [ [ CN9Eps | CN10Eps ] "^N^":0 {pombe} ] |  # pombe ‘beer‘
85      [ [ CN1 | CN2 ] 0:"+Anim+" "^N^":{mw} {anafunzi} ] |  # mwanafunzi ‘student‘
86      [ [ CN5 | CN6 ] 0:"+Anim+" "^N^":0 {shangazi} ] # shangazi ‘aunt‘
87  ] ;
```

A.5 lex/auto-adjs.fsts

```
define Adjs [
    [ CVP "^VADJ^":0 {a} ] |  # a ‘of‘
    [ CVP "^VADJ^":0 {ake} ] |  # ake ‘POSS.3P.Sg‘
    [ CVP "^VADJ^":0 {ako} ] |  # ako ‘your‘
    [ CVP "^VADJ^":0 {angu} ] |  # angu ‘POSS.1P.SG‘
    [ CVP "^VADJ^":0 {ao} ] |  # ao ‘POSS.3P.PL‘
    [ CNP "^NADJ^":0 {baya} ] |  # baya ‘bad‘
    [ CNP "^NADJ^":0 {dogo} ] |  # dogo ‘small‘
    [ CNP "^NADJ^":0 {ema} ] |  # ema ‘good‘
    [ CVP "^VADJ^":0 {enu} ] |  # enu ‘POSS.2P.PL‘
    [ CVP "^VADJ^":0 {enye} ] |  # enye ‘having‘
    [ CVP "^VADJ^":0 {etu} ] |  # etu ‘POSS.1P.PL‘
    [ CNP "^NADJ^":0 {ingine} ] |  # ingine ‘another‘
    [ CNP "^NADJ^":0 {kali} ] |  # kali ‘fierce‘
    [ CNP "^NADJ^":0 {kubwa} ] |  # kubwa ‘big‘
    [ CVP "^VADJ^":0 {le} ] |  # le ‘that‘
    [ CNP "^NADJ^":0 {manjano} ] |  # manjano ‘yellow‘
    [ CNP "^NADJ^":0 {moja} ] |  # moja ‘one‘
    [ (CNP "^NADJ^":0) {ngapi} ] |  # ngapi ‘how.much‘
    [ CVP "^VADJ^":0 {ote} ] |  # ote ‘all‘
    [ CNP "^NADJ^":0 {pumbavu} ] |  # pumbavu ‘foolish‘
    [ CNP "^NADJ^":0 {pya} ] |  # pya ‘new‘
    [ CNP "^NADJ^":0 {refu} ] |  # refu ‘long‘
    [ CNP "^NADJ^":0 {wili} ] |  # wili ‘two‘
    [ CNP "^NADJ^":0 {chache} ] |  # chache ‘few‘
    [ CNP "^NADJ^":0 {zuri} ] # zuri ‘fine‘
] ;
```

A.6 lex/auto-verbs.fsts

```
define VStemBantu [
    {anguk} 0:a |  # anguka ‘fall‘
    {fundish} 0:a |  # fundisha ‘teach‘
    {fik} 0:a |  # fika ‘arrive‘
    {ti} 0:a |  # tia ‘put‘
    {ondok} 0:a |  # ondoka ‘leave‘
    {ib} 0:a |  # iba ‘steal‘
    {imb} 0:a |  # imba ‘sing‘
    {imbi} 0:a |  # imbia ‘sing.for‘
    {ju} 0:a |  # jua ‘know‘
    {ka} 0:a |  # kaa ‘sit‘
    {kuj} 0:a |  # kuja ‘come‘
    "^MONOV^":0 [l] 0:a |  # kula ‘eat‘
    "^MONOV^":0 [e n d] 0:a |  # kwenda ‘go‘
    {none} 0:a |  # nonea ‘kiss‘
    {on} 0:a |  # ona ‘see‘
    {pat} 0:a |  # pata ‘get‘
    {pemb} 0:a |  # pemba ‘trick‘
    {pend} 0:a |  # penda ‘like‘
    {tak} 0:a |  # taka ‘want‘
    {pig} 0:a |  # piga ‘beat‘
    {pik} 0:a |  # pika ‘cook‘
    {siki} 0:a |  # sikia ‘hear‘
    {zoe} 0:a |  # zoea ‘be.accustomed.to‘
    {som} 0:a |  # soma ‘read‘
    {tafut} 0:a |  # tafuta ‘look.for‘
    {tok} 0:a |  # toka ‘leave‘
    {tumi} 0:a |  # tumia ‘use‘
    {waz} 0:a |  # waza ‘ponder‘
    {chez} 0:a |  # cheza ‘play‘
    {kata} 0:a |  # kataa ‘refuse‘
```

```
    {o} 0:a |  # oa ‘marry‘
    {fungu} 0:a |  # fungua ‘open‘
    {pati} 0:a |  # patia ‘get.for‘
    {some} 0:a |  # somea ‘study.for‘
    {andik} 0:a |  # andika ‘write‘
    {nunu} 0:a |  # nunua ‘buy‘
    {lim} 0:a |  # lima ‘cultivate‘
    {kutan} 0:a # kutana ‘meet‘
] ;

define VStemBantuPass [
    "^MONOV^":0 [w] 0:a # kuwa ‘be‘
] ;

define VStemArabicI [
    {da} 0:i |  # dai ‘claim‘
    {fikir} 0:i |  # fikiri ‘think‘
    {furah} 0:i |  # furahi ‘be.happy‘
    {lagha} 0:i |  # laghai ‘trick‘
    {rud} 0:i |  # rudi ‘return‘
    {kubal} 0:i # kubali ‘agree‘
] ;

define VStemArabicE [
    {sameh} 0:e # samehe ‘forgive‘
] ;

define VStemArabicU [
    {saha} 0:u |  # sahau ‘forget‘
    {jib} 0:u # jibu ‘answer‘
] ;
```

A.7 lex/auto-etc.fsts

```
define Etc [
    [ {kama} ] |  # kama (COMP) ‘like‘
    [ {na} ] # na (CONJ) ‘and‘
] ;
```

A.8 lex/misc.fsts

```
define Misc [

    # amba
    [ {amba} VRel ] |

    # naye, nazo, etc...
    [ {na} (VRel) ] |

    # kwa, kwao, kwaye
    [ {kwa} (VRel) ] |

    [ CVP "^VADJ^":0 {a} ] |  # a ‘of‘
    [ CVP "^VADJ^":0 {ake} ] |  # ake ‘POSS.3P.Sg‘
    [ CVP "^VADJ^":0 {ako} ] |  # ako ‘your‘
    [ CVP "^VADJ^":0 {angu} ] |  # angu ‘POSS.1P.SG‘
    [ CVP "^VADJ^":0 {ao} ]   # ao ‘POSS.3P.PL‘

];
```

A.9 rules/nonconcat-pre.fsts

```
read regex [

    # get spaces in to make regex syntax
    [ Letter+ @-> "{" ... "}" || "^[" ?+ _ ?+ "^]"] .o.

    # reconfigures symbols to turn them into multichar symbols in subnet
    [ CTSymbol -> %" ... %" ]


];
```

A.10 lex/nonconcat.fsts

```
read regex [

    # hxxx and hxxo
    [
        [h "^HXX^" "^[" "["]:0 CVH ["]" {^2} "^]"]:0
        [
            # hii, hili, hizi
            [0:{hxxx}] |

            # hayo, hili, hizo, hicho, etc
            [ ["^OREF^" o]:{hxxo} ]
        ]
    ]



];
```

A.11 rules/nonconcat-post.fsts

```
define NonConcatRules [
    # haya not hyaya
    # fixes extra vowels etc
    [ Consonant -> 0 || "^HXX^" _ ]
];
```

A.12 lex/sublex-verbs.fsts

```
# VERB TEMPLATE DEFINITIONS

# Special prefixes only used on verbs
define CV1Sg [ {ni}:["+Anim+" "+Sg+" "+P1+"] ] ;
define CV2SgSubj [ u:["+Anim+" "+Sg+" "+P2+"] ] ;
define CV2SgObj [ {ku}:["+Anim+" "+Sg+" "+P2+"] ] ;
define CV3SgSubj [ [a|{yu}]:["+Anim+" "+Sg+" "+P3+"] ] ;
define CV3SgObj [ [m "^CTMV^"]:["+Anim+" "+Sg+" "+P3+"] ] ;
define CV1Pl [ {tu}:["+Anim+" "+Pl+" "+P1+"] ] ;
define CV2PlSubj [ [m "^CTMV^"]:["+Anim+" "+Pl+" "+P2+"] ] ;
define CV2PlObj [ {wa}:["+Anim+" "+Pl+" "+P2+"] ] ;
define CV3Pl [ {wa}:["+Anim+" "+Pl+" "+P3+"] ] ;
define CV16Obj [ {po}:["+C16+"] ] ;
define CV17Obj [ {ko}:["+C17+"] ] ;
```

```
15 define CV18Obj [ {mo}:["+C18+"] ] ;
16
17 # Define the set of verb prefixes used as subject markers
18 define CVV [
19     [CVP - [CV1 | CV2 ]] |
20     CV1Sg |
21     CV2SgSubj |
22     CV3SgSubj |
23     CV1Pl |
24     CV2PlSubj |
25     CV3Pl
26 ] ;
27
28 # Define the set of verb prefixes used as object markers
29 define CVVObj [
30     [CVV - [CV2SgSubj | CV3SgSubj | CV2PlSubj | CV16 | CV17 | CV18]] |
31     CV2SgObj |
32     CV3SgObj |
33     CV2PlObj |
34     CV16Obj |
35     CV17Obj |
36     CV18Obj
37 ] ;
38
39 # Negative prefixes
40 define VNeg [
41     [
42         [{si}:["+Neg+" "+Anim+" "+Sg+" "+P1+"]] |
43         [{hu}:["+Neg+" "+Anim+" "+Sg+" "+P2+"]] |
44         [{ha}:["+Neg+" "+Anim+" "+Sg+" "+P3+"]] |
45         [{ha}:"+Neg+" [CVV - [CV1Sg | CV2Sg | CV3Sg]]]
46     ] "^NEG^":0
47
48 ];
49
50 # Shorthand for present tense
51 define VTensePres [ {na}:"+Pres+" ] ;
52
53 # Shorthand for 'strong' future tense
54 define VTenseFutStrong [
55     {taka}:"+Fut+"
56 ] ;
57
58 # Define all 'strong' tenses (can take stress)
59 define VTenseStrong [
60     VTensePres |
61     [ ["^PAST^" {li}]:"+Past+" ] |
62     VTenseFutStrong
63 ];
64
65 # Define tenses which can't take stress
66 define VTenseWeak [
67     [ {ta}:"+Fut+" ] |
68     [ ["^PERF^" {me}]:"+Perf+" ] |
69     [ {nge}:["+Pres+" "+Cond+"] ] |
70     [ {ngali}:["+Past+" "+Cond+"] ] |
71     [ {ki}:["+Prog+"] ] |
72     [ {ka}:["+Conc+"] ]
73 ] ;
74
75 # Define 'normal' tenses as everything but 'taka'
76 define VTenseNormal [
```

```
77      [VTenseStrong - VTenseFutStrong] |
78      VTenseWeak
79  ];
80
81  # Shorthand for slot-3 negative
82  define NegSi [
83      {si}:"+Neg+"
84  ] ;
85
86  # Define set of relative markers
87  define VRel [
88      [[CVP - [CV1 | CV2 | CV16 | CV17 | CV18]] ["^OREF^" o]:"+Rel+"] |
89      [[CV16 | CV17 | CV18] ["^OREF^" o]:["+Rel+" "+SemRel+"]] |
90      [{ye}:["+Anim+" "+Sg+" "+P3+" "+Rel+"]] |
91      [{o}:["+Anim+" "+Pl+" "+P3+" "+Rel+"]] |
92      [{vyo}:["+C8+" "+Rel+" "+SemRel+"]]
93  ];
94
95  # Define the set of copular-acting particles
96  define VCop [
97      [{ni}:["+Eps+"]] |
98      [CVV - CV3SgSubj] |
99      [{yu}:["+Anim+" "+Sg+" "+P3+"]]
100 ];
101
102 # Group arabic stems together
103 define StemAndFVArabic [
104     [VStemArabicI i:0] |
105     [VStemArabicE e:0] |
106     [VStemArabicU u:0]
107 ];
108
109 # combine passive forms, don't include VStemBantuPass
110 define PassiveStem [
111     [VStemBantu | [StemAndFVArabic "^ASTEM^":0]] ["^PASS^" w]:"+Pass+"
112 ];
113
114 # Group all stems together
115 define StemAndFV [
116     [ [ VStemBantu | VStemBantuPass | PassiveStem ] a:0] | StemAndFVArabic
117 ];
118
119 # Define the templates
120 define Verbs [
121
122     # Basic positive imperative with optional object
123     # (i)chukua! 'take (it)!'
124     [ (CVVObj) StemAndFV 0:"+Imper+" ] |
125
126     # pos + neg subjunctive, optional object
127     # u(si)(i)chukue! 'you should (not) take (it)'
128     [ CVV (NegSi) 0:"+Notense+" (CVVObj) [["^MONOVOK^":0 VStemBantu e:0] |
129       StemAndFVArabic] 0:"+Subjunct+" ] |
130
131     # subj pronoun, tense, optional obj pronoun, stem+fv
132     # e.g. nina(i)pika 'i am cooking (it)'
133     [ CVV VTenseNormal (CVVObj) StemAndFV ] |
134
135     # same as above with infixed relative pronoun (restrictions on tense)
136     # tense can be timeless negative 'si'
137     # e.g. aliyefundisha 'she who taught', kisichofaa 'things which are not suitable'
138     [ CVV [VTenseStrong | [NegSi 0:"+Notense+"]] VRel (CVVObj) StemAndFV ] |
```

```
139
140     # negative subj pronoun, tense (not pres), optional obj pronoun
141     # e.g., haku(i)pika ‘he did not cook (it)‘
142     [ VNeg [VTenseNormal - VTensePres] (CVVObj) StemAndFV ] |
143
144     # same as above, but in present tense
145     # e.g., ha(i)piki ‘he isn’t cooking (it)‘
146     # (final vowel a->i in Bantu verbs)
147     [ VNeg 0:"+Pres+" (CVVObj) [StemAndFVArabic |
148       ["^MONOVOK^":0 VStemBantu i:0]] ] |
149
150     # copular verbs (either ’ni’ or verb class prefix)
151     [ VCop 0:"+Cop+" ] |
152
153     # copular relatives (say are underlingly ’kuwa’)
154     [ CVV {li}:"+Cop+" VRel ] |
155
156     # ’general relative’ type
157     # asemavyo ‘he who ...’
158     [ CVV (NegSi) 0:"+Notense+" (CVVObj) StemAndFV VRel ] |
159
160     # infinitives
161     [ {ku}:["+Inf+"] (CVVObj) StemAndFV ]
162 ];
```

A.13 rules/phonology.fsts

```
1   define Phonology [
2
3       # epenthesizes a ’w’ in between ’m’ and vowel
4       # mema -> mwema, mili -> mwili, alimona -> alimwona
5       [ [..] -> w || m [["^CTM^" NA] | "^CTMV^"] _ Vowel ] .o.
6
7       # deletes extra ’i’ in ’ii’
8       # kiingi -> kingi
9       [ i -> 0 || i CTI NA _ ] .o.
10
11      # class 5/6eps (no ’ji’ in front)
12      [ j i -> 0 || _ "^CTI5^" NA SyllCons Syll ] .o.
13
14      # bleed palatalization of ’i’, instead assimilate following V for class 5/6
15      # but only for nouns!
16      # jiiko -> jiko (’iko’ postulated because of pl ’meko’ < ’ma-iko’)
17      [ Vowel -> 0 || i "^CTI5^" "^N^" _ ] .o.
18
19      # palatalization of ’i’ before all Vowels in initial contexts
20      # ia -> ya
21      [ i -> y || .#. _ [ [CTI NA] | ["^CTIV^" "^VADJ^"]] Vowel ] .o.
22
23      # palatalization of ’i’ after ’m’ before ’e’, but only for adjs
24      # miema -> myema, iliio -> iliyo
25      # NOT miezi -> myezi
26      [ i -> y || m _ CTI "^NADJ^" e ] .o.
27
28      # deletion of ’i’ after ’j’ before ’e’
29      # jiema -> jema
30      [ i -> 0 || j _ CTI NA e ] .o.
31
32      # class 7/8 palatalization before vowels, including verb agreement
33      # kieti -> cheti, kio -> cho
34      [ [k i [ [CTI NA]|["^CTIV^"["^VADJ^"|"^OREF^"]]]] -> [c h] || _ Vowel ] .o.
```

```
35      # vieti -> vyeti
36      # vio -> vyo
37      [ [v i] -> [v y] || _ [[CTI NA]|["^CTIV^"["^VADJ^"|"^OREF^"]]] Vowel ] .o.
38
39      # for verb agreement adjs only, delete 'i' after cons, before vowel
40      # (bled by 7/8 palatalization, otherwise we'd get kia -> ka !-> cha)
41      # lia -> la, zia -> za, lienye -> lenye
42      [ i -> 0 || .#. Consonant _ "^CTIV^" "^VADJ^" Vowel ] .o.
43
44      # class 9/10 ni-stuff
45      # take out 'i' before consonant, so 'nivua' -> 'nvua' (feeds n->m)
46      [ i -> 0 || n _ "^CTNI^" NA Consonant ] .o.
47      # ni-umba -> ny-umba, ni-avu -> ny-avu
48      [ i -> y || n _ "^CTNI^" NA Vowel ] .o.
49      # ny-ema -> njema
50      [ y -> j || n _ "^CTNI^" NA e ] .o.
51      # nsauti -> sauti, but not npya -> pya
52      [ n -> 0 || _ "^CTNI^" NA VoicelessOnset Nucleus Syll+ ] .o.
53      # nvua -> mvua
54      [ n -> m || _ "^CTNI^" NA Labial ] .o.
55      # mwili -> mbili
56      [ w -> b || "^CTNI^" NA _ ] .o.
57      # nlimi -> ndimi, nrefu -> ndefu
58      [ [l | r] -> d || "^CTNI^" NA _ ] .o.
59
60      # 'a'-contraction
61      # waingine -> waengine (to feed next rule)
62      [ i -> e || "^CTA^" NA _ ] .o.
63      # waengine -> wengine, waema -> wema, maeko -> meko, maaasi -> maasi
64      # BUT NOT maembe -> membe: taken care of by epsilon group in 5/6
65      [ a -> 0 || _ "^CTA^" NA [e|a] ] .o.
66      # yaa -> ya, waalimu -> walimu (removal of double 'a' in cl1/2 and v-adj
67      # agreement)
68      [ a -> 0 || _ [ ["^CTA^" NA] | ["^CTAV^" "^VADJ^"]] a ] .o.
69
70      # consonantalization of 'u' in class 13/14, tu-, u-
71      # for nouns: uimbo -> wimbo, but NOT uovu -> wovu
72      # for adjs w/ noun agreement: uingine -> wingine
73      # for adjs w/ verb agreement: ua -> wa, ALSO uote -> wote
74      [ u -> w || _ [ ["^CTU^" NA [Vowel - o]] | ["^CTUV^" "^VADJ^" Vowel]]] .o.
75
76      # relative -o (leftover cases from above)
77      # wao -> o
78      # bleeds next rule
79      [ [w a] -> 0 || _ "^CTAV^" "^OREF^" ] .o.
80      # pao -> po
81      [ a -> 0 || _ "^CTAV^" "^OREF^" ] .o.
82      # uo -> o
83      [ u -> 0 || _ "^CTUV^" "^OREF^" ] .o.
84      # lio -> lo
85      # this needs to bleed next rule+1
86      [ [l i] -> l || _ "^CTIV^" "^OREF^" ] .o.
87      # zio -> zo
88      # this needs to bleed next rule
89      [ [z i] -> z || _ "^CTIV^" "^OREF^" ] .o.
90      # io -> yo
91      [ i -> y || _ "^CTIV^" "^OREF^" ]
92
93  ];
```

A.14 rules/verbal-morph.fsts

```
define VerbalMorph [

    # take care of silipiga -> sikupiga
    [ [l i] -> [k u] || "^NEG^" "^PAST^" _ ] .o.

    # take care of simepiga -> sijapiga
    [ [m e] -> [j a] || "^NEG^" "^PERF^" _ ] .o.

    # take care of monosyllabic verbs
    # first, insert ku on all monosyllabic verb stems
    [ [..] -> [k u] || "^MONOV^" _ ] .o.
    # now find the pattern monovok-monov-ku and delete anywhere
    # essentially, take ku back again if in a ku-ok context
    [ ["^MONOVOK^" "^MONOV^" k u] -> 0 || _ ] .o.
    # deal with ku->kw
    [ u -> w || "^MONOV^" k _ Vowel ] .o.

    # passive stuff
    # katawa -> kataliwa, fungua -> funguliwa
    [ w -> {liw} || [a|u] "^PASS^" _ a ] .o.

    # owa -> olewa
    [ w -> {lew} || o "^PASS^" _ a ] .o.

    # sahauwa -> sahauliwa
    # bleeds next rule
    [ w -> {liw} || a u "^ASTEM^" "^PASS^" _ a ] .o.

    # jibuwa -> jibiwa
    [ u -> i || _ "^ASTEM^" "^PASS^" w a ]

] ;
```

A.15 rules/irregulars.fsts

```
define Irregulars [

    # fix irregular njema
    # previous rule: nema -> nyema
    # this rule: nyema -> njema
    [ y -> j || _ "^CTNI^" "^NADJ^" e m a .#.] .o.

    # fix 3/4 nouns which don't epenthesize 'w'
    # mwoto -> moto, mwoshi -> moshi
    [ w -> 0 || m "^CTM^" "^N^" _ o [{to} | {shi} | {yo}] .#.] .o.

    # fix 3/4 nouns which epenthesize 'u'
    # mda -> muda, mhogo -> muhogo
    [ [..] -> u || m "^CTM^" _ "^N^" [{hogo} | {da} | {wa}] .#. ] .o.

    # fix irregular mume
    # mwume -> mume
    [ w -> 0 || m "^CTM^" "^N^" _ u m e .#. ] .o.

    # fix irregular wanaume
    # wanawaume -> wanaume
    [ [w a] -> 0 || w "^CTA^" "^N^" a n a _ u m e .#. ] .o.

    # fix proper names
```

```
25      [ m -> 0 || _ "^CTM^" "^PN^" ]
26  ];
```

A.16 rules/cleanup.fsts

```
1  # Clean up is nice and easy since everything has been defined!
2
3  define Cleanup [ Symbol -> 0 ] ;
```