Temiar Reduplication in One-Level Prosodic Morphology

Markus Walther
University of Marburg
FB09/IGS, Wilhelm-Röpke-Str. 6A, D-35032 Marburg, Germany
Markus.Walther@mailer.uni-marburg.de

Abstract

Temiar reduplication is a difficult piece of prosodic morphology. This paper presents the first computational analysis of Temiar reduplication, using the novel finite-state approach of One-Level Prosodic Morphology originally developed by Walther (1999b, 2000). After reviewing both the data and the basic tenets of One-level Prosodic Morphology, the analysis is laid out in some detail, using the notation of the FSA Utilities finite-state toolkit (van Noord 1997). One important discovery is that in this approach one can easily define a regular expression operator which ambiguously scans a string in the left- or rightward direction for a certain prosodic property. This yields an elegant account of base-length-dependent triggering of reduplication as found in Temiar.

1 Introduction

Temiar is an Austroasiatic language of the Mon-Khmer group spoken by a variety of tribal people in West Malaysia (Benjamin, 1976). Its intricate morphological system has received some attention in the theoretical literature. The main focus has been on the aspectual morphology of verbs, where an interesting pattern of partial reduplication emerges that is sensitive to the size of the verbal root. For example, in the active continuative, *gɛlgəl* 'to eat' reduplicates both the initial /g/ and the final /l/ of its monosyllabic base *gəl*. In contrast, bisyllabic *səluh* 'to shoot' comes out as *sɛhluh*, where only the final /h/ is copied, this time as an infix.

Temiar reduplication thus appears to be a suitably rich testing ground for a novel approach to reduplication developed by (Walther, 1999b; Walther, 2000) within a finite-state framework. Even though that approach, One-Level Prosodic Morphology, was presented from the outset as being generally applicable, it has been proven time and time again that only concrete empirical application of a particular approach to computational morphology and phonology will fully reveal its inherent virtues and weaknesses. As an example, (Beesley, 1998) reports that it was actual experimentation with grammars of word-formation in Arabic and Hungarian which fully revealed the negative effects of modelling long-distance circumfixional dependencies in purely finite-state terms, subsequently leading to some suggestions for improvement.

It is perhaps worth emphasizing that (Walther, 1999b)'s solution for reduplication in a finite-state context is preferrable for cross-linguistic validation precisely because it is the first that solves the problem in the *general* case. Because reduplication often involves copying of a strictly bounded amount of material, the bounded case *could* in principle be modelled as a finite-state process by enumerating all possible forms of the copy and then making sure each was matched to the proper stem. To solve this simplified problem, no new techniques are needed in theory. In practice however, the brute-force enumeration approach apparently has not been pursued further, apart from isolated examples (see Antworth (1990), p.157f for a fixed-size case in Tagalog). This is probably because such an approach is awkward to specify in actual grammars and because it will inevitably lead to an explosion of the state space (Sproat (1992), p.161). Finally and in contrast to (Walther, 1999b), it would clearly break down for *productive* total reduplication, which is isomorphic to the context-sensitive language $\{ww | w \in \Sigma^+\}$.

A second motivation for choosing Temiar is that all prior analyses of its data are heavily underformalized and incomplete, irrespective of whether they are situated in the older rule paradigm (McCarthy, 1982; Broselow and McCarthy, 1983; Sloan, 1988; Shaw, 1993) or an optimality-theoretic setting (Gafos, 1995; Gafos, 1996; Gafos, 1998b; Gafos, 1998a). Hence a formalized and computationally tested analysis that strives to keep a healthy balance with respect to linguistic adequacy would represent significant progress on its own.

In the rest of the paper I will attempt to provide just such an analysis, beginning in §2 with a presentation of the relevant data. Next, section §3 reviews the core of One-Level Prosodic Morphology, which will be used as formal background. Using that background, the analysis is then fully developed in §4. The paper concludes with some discussion in §5.

2 Temiar reduplication

All data on Temiar reduplication in this section come from (Benjamin, 1976), the main source on the subject.[1] According to Benjamin, the characteristic aspectual paradigms of "monosyllabic and schewa-form verbs" (B:168) are as follows (B:169):

(1)	‘to call’	‘to lie down/sleep/marry’	
	‘monosyllabic’	‘schewa-form’	
active	ˈkɔɔw	sə.ˈlɔg	perfective
	ka.ˈkɔɔw	sa.ˈlɔg	simulfactive
	kɛ**w**.ˈkɔɔw	sɛ**g**.ˈlɔg	continuative
causat.	tɛr.ˈkɔɔw	sɛr.ˈlɔg	perfective
	tə.ra.ˈkɔɔw	sə.ra.ˈlɔg	simulfactive
	tə.rɛ**w**.ˈkɔɔw	sə.rɛ**g**.ˈlɔg	continuative

We have inferred syllabifications in (1) from the statement that "only two types of syllables occur: *open syllables* of canonical form CV, and *closed syllables* of canonical form CVC" (B:141). Note that Benjamin abstracts from vowel length here. Word-level stress, which is "falling regularly on the final syllable" (B:139), is likewise inferred in (1). Observe that only monosyllabic roots like *kɔɔw* reduplicate their initial consonant in the non-perfective aspectual forms of the active, while longer roots like *səlɔg* do not. This contrasts with obligatory reduplication of the root-final consonant in the continuative.

An important further generalization is that all extra segmental material beyond the bare root is inserted immediately before the stressed syllable, leading to prefixation for monosyllabic roots, but infixation in polysyllabic ones (Gafos, 1998b). From this point of view we can also see a correlation between the fact that causative forms of monosyllabic roots – which must be at least bisyllabic – begin with a fixed /t/[2] and the restriction that words must "always begin and end with a consonant" (B:141). In triconsonantal roots like *səlɔg* that restriction is taken care of by the first root consonant itself, so no fixed segment needs to appear.

According to Benjamin, prefinal syllables – which are unstressed – can show alternation of their vocalic quality: "In prefinal closed syllables the inner vowels /e ə o/ are replaced by the outer vowels /i ɛ u/ respectively" (B:144). This descriptive generalization accounts for the remaining contrasts in (1), witness e.g. *sə.lɔg* versus *sɛg.lɔg*.

It is interesting to see that Temiar even exhibits phonological modifications between base and reduplicant, affecting consonants in the continuative:

(2)				
	yaap	→	**yɛm**.yaap	‘to cry’ (B:143)
	pət	→	**pɛn**.pət	‘to long for’ (B:146)
	sə.lɔɔk	→	sɛ**ŋ**.lɔɔk	‘to hunt successfully’ (B:146)

Benjamin explains that medial coda consonants from the class of oral voiceless stops turn into their voiced nasal equivalents in Northern Temiar (and to plain voiced stops in the Southern dialect; B:143).

It is of some importance to clarify a number of further aspects of the data and their interpretation. First, theorists have frequently employed the stronger term ‘minor syllables’ for Benjamin's prefinal syllables, reflecting their alleged special status by means of an impoverished representation (e.g. empty syllable nuclei in (Gafos, 1998b)) and/or further formal mechanisms (e.g. a ban on full vowels in prefinal position *PREFINAL-V (Gafos, 1998a)). We do not follow this move here, because empirically it is neither true that penultimate vowels are categorically restricted to schwa-like vowels (*halab* ‘to go downriver’, *sindul* ‘to float’, etc.) nor are there any solid statistics of a presumed tendency to vowel reduction in unstressed syllables, nor can the variable quality of prefinal vowels be consistently derived from flanking consonants. Hence, such penultimative vowels are to be lexically specified as alternating.

Second, Benjamin's subclass restriction of (1) to "monosyllabic and schewa-form verbs" correctly excludes polysyllabic roots like the already mentioned *halab* and *sindul*, where prefinal open syllables with vowels outside of /e ə o/ occur. These roots undergo "very few morphological changes" (B:170), basically proclitization.

[1]We will abbreviate further references to this work with "(B: <page number>)" in the text. Moreover, to highlight reduplicated parts in the data they will often be printed in bold.

[2]Or /b/, if the root starts in /c,t/: /caaʔ/ ‘to eat’ gives /bɛr.caaʔ/ ‘to feed’ (B:169).

Third, paradigms for a given root are hardly ever complete, with various irregularities and non-productive patterns also occuring (B:169f). Again, a good deal of lexicalization would seem necessary to correctly describe Temiar verbs in a realistic grammar fragment.

Given this descriptive summary, our goals for the upcoming analysis are, first, to treat the *full* paradigm of (1). As a second goal, we would like to reflect the emergent formal desiderata in a transparent way, in particular referring to the need to account for *repetition, truncation, infixation* and *phonological modification*. Thirdly, we will attempt a *compositional* analysis of the morphological exponency of aspect.

3 One-Level Prosodic Morphology

In order to provide the necessary background for the Temiar analysis in §4, this section briefly reviews the finite-state approach to prosodic morphology developed in (Walther, 1999b),

That work itself was presented as an extension to (Bird and Ellison, 1994)'s One-Level Phonology framework, where phonological representations, morphemes and more abstract generalizations are all finite-state automata that express surface-true constraints on word forms, and constraint combination is by automata intersection.

In a nutshell, the extension comprises three main components. We (i) represent phonological strings differently for purposes of modelling prosodic morphology, (ii) implement reduplicative coyping by automata intersection, and (iii) introduce a resource-conscious variant of automata.

For (i), operators are provided that construct enriched automata from a simple string automaton, in particular giving it a kind of doubly-linked structure so that the symbol repetition inherent in reduplication translates into following backwards-pointing technical transitions. The individual enrichments involve only local computation per state or transition, so that on-the-fly implementation is easy if desired. In other words, one does not necessarily have to enrich the entire lexicon in advance.

Enriched representations In a bit more detail, the enrichments of (i) are as follows. The three aspects of *reduplication* or symbol repetition, *truncation* or symbol skipping and *infixation* or transitive, non-immediate precedence of symbols are reflected in three regular expression operators, $add\_repeats, add\_skips, add\_self\_loops$. Each takes the underlying automaton A of a regular language L_A as its only argument. Formally, they can be defined as follows:

(3) Let $A = (Q, \Sigma, \delta, q_0, F)$ be the minimal ϵ-free[3] finite-state automaton for L_A, with Q a finite set of states, finite alphabet Σ, transition function $\delta : Q \times \Sigma \mapsto 2^Q$, start state $q_0 \in Q$ and set of final states $F \subseteq Q$.

a. Assume $repeat \notin \Sigma$.
$add\_repeats(A) \overset{def}{=} (Q, \Sigma', \delta', q_0, F)$, where $\Sigma' = \Sigma \cup \{repeat\}$,
$\forall x \in \Sigma\, \forall q \in Q$: $\delta'(q, x) = \delta(q, x)$ and
$\forall p \in Q$:$\delta'(p, repeat) = \{q | \, p \in \delta(q, x)\}$

b. Assume $skip \notin \Sigma$.
$add\_skips(A) \overset{def}{=} (Q, \Sigma', \delta', q_0, F)$, where $\Sigma' = \Sigma \cup \{skip\}$,
$\forall x \in \Sigma\, \forall q \in Q$: $\delta'(q, x) = \delta(q, x)$ and
$\forall q \in Q$: $\delta'(q, skip) = \delta(q, x)$

c. $add\_self\_loops(A) \overset{def}{=} (Q, \Sigma, \delta', q_0, F)$, where
$\delta' = \delta \cup \{(q, \sigma, \{q\}) | \, q \in Q, \, \sigma \in \Sigma\}$

An example enrichment of Temiar *səlɔg* is shown in figure 1. One can imagine how *skip* and *repeat* transitions allow, figuratively speaking, forward and backward movement within a string, while self loops will absorb infixal morphemes that are intersected with fig. 1. Finally, so-called *synchronization bits* **:1, :0** were introduced in (Walther, 1999b) to define the extent of a reduplicative base constituent in a segment-independent way. Bit value **:1** marks the edges and **:0** the interior segments of a base, as shown in fig. 1 for a hypothetical whole-root reduplication pattern. In actual practive, synchronization

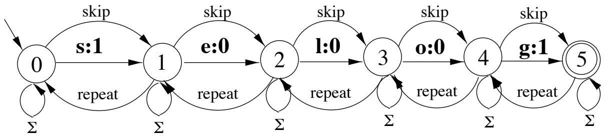

Figure 1: add_repeats(add_skips(add_self_loops(selog)))

bits are sets of symbols, just like the rest of the alphabet. Sets as transition labels improve over traditional automata in terms of automata compactness, were already proposed for phonology in (Bird

[3]Minimality prevents non-(co)-accessible transitions from getting enriched, while lack of ϵ transitions keeps positional $skip/repeat$ 'movement' in lockstep with segmental positions.

and Ellison, 1992) and do not increase mathematical expressivity beyond regular languages.[4] Hence, the segmental part of fig. 1 may be defined in a modular fashion through the intersection of strings of symbol sets that mention only certain dimensions (here: phonemes and synchronisation bits), being underspecified for the unmentioned dimensions. We will again follow (Walther, 1999b) in conceiving of sets as types arranged in a type hierarchy that is structured by set inclusion, and also in allowing arbitrary boolean combinations of types.

Copying as intersection Given enriched representations as in fig. 1, various patterns of reduplication are now easy to define. We can denote a synchronised abstract string by the regular expression

$$base \equiv seg{:}1\, seg{:}0^{*}\, seg{:}1$$

where seg is the type subsuming all phonological segments. Then hypothetical total reduplication – unattested in Temiar, but wellknown from Indonesian and many other languages – is described by

$$total \equiv base\, \boldsymbol{repeat}^{*}\, base$$

A variant slightly more akin to Temiar – and actually attested in the neighbouring language Semai – that $skip$s the interior of the base in a prefixed reduplicant is just as easy:

$$semai \equiv seg{:}1\, \boldsymbol{skip}^{*}\, seg{:}1\, \boldsymbol{repeat}^{*}\, base$$

Ignoring self loops for the moment, all we need now to apply a reduplication pattern to an enriched base representation is simply to *intersect* the former with the latter: automata intersection has sufficient formal power to implement reduplicative copying! Here is an example, using the abbreviation $selog \equiv \boldsymbol{s{:}1\, e{:}0\, l{:}0\, o{:}0\, g{:}1}$ for perspicuous display:

$$add\_repeats(selog) \cap total \equiv selog\, \boldsymbol{repeat^{5}}\, selog$$

As pointed out in (Walther, 2000), generalizing to a *set* of bases involves nothing more than enriching each base separately, then forming the union of the resulting automata. The opposite order would produce unwanted cross-string repetition, since $add\_repeats$ does not distribute over union. However, an unpublished experiment shows that on-demand implementation of a slightly modified $add\_repeats$ can help to preserve the memory efficiency of building a minimized base lexicon as the union of individual base strings first. Due to lack of space, the details will be reported elsewhere.

Resource consciousness As much as we need the formal means provided by self loops for infixations like Temiar *s-a-lɔg*, the resulting automata overgenerate massively. What's missing according to (Walther, 1999b) is a distinction between explicitly contributed, independent information (e.g. the infix *-a-* itself) and contextual, dependent information that is tolerated but must be provided by other constraints (e.g. the $1 \overset{\Sigma}{\to} 1$ self loop that *hosts* the infix). Therefore, a parallel distinction between two kinds of symbols – producers and consumers – was introduced. In that scenario a symbol represents an information resource that needs to be produced at least once, then can be consumed arbitrarily often. To utilize the distinction, an additional P/C bit accompanies symbols, with P/C = 1 for producers. All symbols introduced by the three enrichment operators are consumers. Furthermore, automata intersection is made aware of these resource-conscious notions by splitting it into two variants: In open interpretation mode, P/C bits of matching symbols are combined by logical OR, so that a result transition will be marked as a producer whenever at least one argument transition is a producer. In closed interpretation mode, combination is by logical AND instead, allowing only producer-producer matches. Grammatical evaluation can then be characterized as follows:

$$(\text{Lexicon} \cap_{open} \text{Constraint}_1 \cdots \cap_{open} \text{Constraint}_N) \cap_{closed} \boldsymbol{\Sigma}^{*}$$

Here and elsewhere, producers are in bold print. Note the final intersection with the universal producer language, which eliminates unused consumer transitions, the main source of overgeneration.

4 The analysis

We have assembled enough background now to proceed to the actual analysis of the Temiar data in (1). The analysis is implemented using FSA Utilities, a finite-state toolbox written in Prolog which encourages rapid prototyping (van Noord, 1997). Figure 2 shows a relevant fragment of its syntax (extensions and modifications in italics).

In displaying the grammar, we will take liberty in suppressing certain definitions in the interest of conciseness, relying on the mnemonic value of

[4]Of course, the identity requirement for matching transitions in traditional automata intersection must be replaced by a non-empty intersection requirement for set-based matching.

`{}`	empty language
`[E1,E2, ... ,En]`	concatenation
`{E1,E2, ... ,En}`	union
`E*`	Kleene closure
`E^`	optionality
`E1 & E2`	intersection
`A` –*l*–> / –*r*–> `( B / C)`	monotonic rules
∼`S`	set complement
Head(arg1, ... , argN) := Body	macro def.

Figure 2: Regular expression operators

their names instead. A case in point is **`producer(T)`**, **`consumer(T)`**: since the names are self-explanatory, it suffices to note that the only argument **`T`** contains type formulae that denote the symbol sets, as explained before. Allowable type-combining operators are conjunction **`&`**, disjunction **`;`** and negation ∼. The same goes for monotonic rules, which – unlike rewrite rules – can only specialize their focussed segmental position **`A`** to **`B`**. They exist in two variants, where **`A -r-> B/C`** notates the case where context **`C`** is right-adjacent to the focus ($A \to B/\_\_C$), and vice versa for **`A -l-> B/C`**.

Syllabification To define the reduplicant in prosodic terms later on, we need **`syllabification`** in the first place. Here a simplified finite-state version of a proposal by (Walther, 1999a) is employed. Its key idea is to allow incremental assignment of syllable roles to segmental positions via a featural decomposition of the three traditional roles, using two binary-valued features **`ons`** and **`cod`**:

(4)

Onset	**`ons`**	∼**`cod`**
Nucleus	∼**`ons`**	∼**`cod`**
Coda	∼**`ons`**	**`cod`**
Coda**O**nset	**`ons`**	**`cod`**

As a side-effect, one gets the fourth role **CO**, a monosegmental prosodic representation of true geminates. The subcomponent **`sbs`**, for sonority-based syllabification, itself rests on the computation of **`sonority_differences`** between adjacent segmental positions (not shown), where sonority may either go **`up`** or **`down`**. Together with some self-explanatory constraints **`obligatory_wordinternal_onsets`** and **`no_geminates`**, prosodic surface wellformedness is then welldefined. Only **`if_doubly_synced_edge_then_stressed`** may seem slightly odd, since it has a purely technical character: it rules out certain illformed alternatives in wordforms. Note, however, that the necesssity of such technical constraints, which are certainly implicit in informal analyses as well, can only be reliably detected in computerized analyses such as the present one, which allow for mechanical enumeration of a grammar's denotation.

```
sbs := [ {  [consumer(down&~ons),
             consumer(segment&~'Nuc')],
            [consumer(up&~'Nuc'),
             consumer(segment&~cod)
         } *, no_final_onset ^].

no_initial_coda := consumer(segment&~cod).
no_final_onset := consumer(segment&~ons).

syllabification := sonority_differences&
 sbs&[no_initial_coda, sbs].

% -- further constraints ---
obligatory_wordinternal_onsets :=
   ( segment -r-> ons / 'Nuc' ). % _ 'N'

no_geminates := consumer(~'CO')*.

prosodic_constraints := obligatory_word-
 internal_onsets & no_geminates &
 if_doubly_synced_edge_then_stressed.

if_doubly_synced_edge_then_stressed :=
 [( {consumer(~':1'),
    [consumer(':1'),consumer(~':1')],
    [consumer(':1'),consumer(':1'),
     consumer(stressed)]
    } *), consumer(':1') ^].
```

Stress Given the assignment of syllable roles to segmental positions, we are now ready to define Temiar word **`stress`**. A possibly empty sequence of **`prefinal_syllables`**, each of which is constrained to be of shape $ON(C)$ and **`unstressed`**, is followed by a final **`stressed syllable`**. The macro **`ends_before_last_syll`** makes sure that the dividing line between the penultimate and ultimate syllable is drawn correctly.

```
stress := [prefinal_syllables &
           ends_before_last_syll,
           syllable].

prefinal_syllables :=
   ([consumer('Ons'), consumer('Nuc'),
    (consumer('Cod') ^) ]*) &
    consumer(unstressed)*.

ends_before_last_syll:=([consumer(segment)*,
                 consumer(segment&~ons)]^).

syllable := [consumer(ons)+,consumer('Nuc'),
             consumer(cod)*] &
            (consumer(stressed)*).
```

Stems We proceed towards the definition of a **stem** by noting that – as described in §2 – both the extent of a **base**'s phonological material *and* its stress pattern are necessary prior knowledge for adding aspectual morphemes in the appropriate way. Hence, we impose the respective constraints onto the *isolated base string* in **stem0**, before wrapping the result in the usual enrichments. However, the addition of self loops for infixation this time is *a priori* restricted to the position immediately before a stressed onset, in accordance with the descriptive generalization stated in §2. Experiments have shown that using the unrestricted *add_self_loops* of (3.c) would cause much unnecessary hassle in *a posteriori* restriction of the possible infix locations to the actually attested ones. It thus appears that Temiar provides a first case for further parametrization of at least one of the original operators from (Walther, 1999b):

```
base := [consumer(':1'),consumer(':0')*,
         consumer(':1')].

stem0(StemMaterial) :=
 add_self_loop_before(stressed&'Ons',
  add_repeats(add_skips(StemMaterial &
   base & syllabification &
   prosodic_constraints & stress))).

stem(Segments) :=
 stem0(stringToSegments(Segments)).
```

Definitions for the actual stem entries of **selog, koow, yaap** are shown below, using the ASCII-IPA mapping {`@` ↦ ə, `E` ↦ ɛ, `O` ↦ ɔ}. In evaluating the first entry, the schwa actually translates into a producer-type disjunction (**ə;ɛ**) with the help of **stringToSegments**. It thus makes sense to constrain this free alternation further, which is the purpose of **has_prefinal_syllable**. While the monosyllable **koow** needs no extra treatment, **yaap** is an example of a stem ending in an **alternating_labial**, whose definition however is straightforward (**medial, final** refer to a positional classification of the word that is defined later):

```
selog := stem("s@lOg") &
       has_prefinal_syllable.

koow := stem("kOOw").
yaap := stem0([stringToSegments("yaa"),
              alternating_labial]).

alternating_labial := {producer(p&final),
                   producer(m&medial&cod)}.
```

If we now define **has_prefinal_syllable** itself, we have completed the components that make up **stem**. While the definition really targets the prefinal vowel, its preceding onset and the stretch of arbitrary material after it must also be mentioned. To tolerate interspersed **technical_symbols**, the **ignore** operator is used (Kaplan and Kay, 1994).

The purpose of **prefinal_V** is to control the alternation between 'outer' and 'inner' vowel, here parametrized for ɛ∼ə only. It does so by referencing the next syllable role: if it is consistent with **ons**, that vowel resides in an open syllable, hence the **close_mid** variant (ə) will be selected. Two elsewhere cases deal with closed syllables and the possible presence of a technical symbol:

```
has_prefinal_syllable :=
  ignore([consumer('Ons'),
          prefinal_V(('E';'@'),
              ':0'&unstressed),
          consumer(anything) *],
          technical_symbols).

technical_symbols :=
 (consumer((skip;repeat)) *).

prefinal_V(Quality, Common) :=
 { [producer(Quality&close_mid&Common),
    consumer(ons)],
   [producer(Quality&~close_mid&Common),
    consumer(cod)],
   [consumer((skip;repeat))]
 } ).
```

Aspectual affixes It is time to concentrate on the most interesting part, and that is how to define the affixes. Again the general picture will be to see them as constraints on word forms which are imposed by intersection. We begin with the **simulfactive**. The claim here is that its characteristic pattern is the realization of the initial base segment (**:1**), followed by the infixed melodic element /a/, and then the entire string that begins with the stressed onset. Phrasing the pattern this way already suffices to capture the difference in reduplication behaviour between *'kɔɔw* and *sə'lɔg*: if we have inserted the *-a-* after the initial consonant in the first base, the stressed onset is *to the left of /a/'s position*, whereas in the second base that onset is found *to the right*. Thus, repetition of segments is necessary to avoid ungrammaticality due to constraint violation in the first case (*k-a-'kɔɔw*), but not in the second (*s-a-'lɔg*).

This behaviour is most naturally modelled by defining a new operator **seek(X)**, which allows for ambiguous movement *either* to the left (**`repeat`**) *or* to the right (**`skip`**) before imposing the restriction **`X`**. This operator is applied to infixal /a/ because it is precisely the infix which needs to 'seek' its prosodically defined unique insertion point, i.e. self loop. Finally, to ensure that the other aspectual morphemes can play their part later on, the entire pattern is wrapped in **`align`** to tolerate further material before (**`align_right`**) and after it (**`align_left`**):

```
simulfactive :=
  align([consumer(':1'),
         seek([producer(a&':0'&unstressed),
         consumer(stressed&'Ons')])]).

seek(X) :=
 [{producer(skip)*,producer(repeat)*},X].

align_left(X):=[X,consumer(anything)*].
align_right(X):=[consumer(anything)*,X].
align(X) := align_right(align_left(X)).
```

Moving on to the **`continuative`**, we can see that the relevant formal generalization is a bit more complex. Again we start off with the initial base segment (**`:1`**), but then seek a place to infix the constant /ε/, before we **`skip_to`** the next synchronised base position (**`:1`**), which inevitably will be the final one. The pattern is completed by again seeking the stressed onset, from which realization of the string proceeds uninterrupted due to the licensing of extra material that the **`align`** wrapper provides. This produces a similar contrast with respect to (non-)reduplication of the first base position, but makes both the repetition of the last base segment and the ~~truncation~~ of its interior material obligatory in both base types (*k-ε- ~~oo~~ w'kɔɔw* vs. *s-ε- ~~lo~~ g'lɔg*):

```
continuative :=
 align([consumer(':1'),
    seek([producer('E'&':0'&unstressed)]),
       skip_to(consumer(':1')),
       seek(consumer(stressed&'Ons'))]).

skip_to(X) := [producer(skip)+, X].
```

What is left now is the proper definition of the **`causative`**. Here we observe from (1) that the causative morphology always starts word-initial, hence the use of **`align_left`**. We have a default consonant /t/ whose realization we must somehow force in the monosyllabic roots. Next comes a vowel, whose quality – ə or ε – is again regulated by the familiar **`has_prefinal_syllable`**. Finally, the characteristic fixed element /r/ is specified. Upon second thought, the /t/ is guaranteed to appear in monosyllable roots, because prefinal syllables always require an onset. The default absence of the /t/ – when not needed on prosodic grounds – is again encoded by the producer/consumer distinction, which contrasts the two disjuncts of the parametrized macro **`default`**:

```
causative :=
 align_left([default(t&unstressed,':1'),
            producer(vowel),
            producer(r&':1'&unstressed)])&
           has_prefinal_syllable.

default(Optional, Common) :=
  { producer(Common&Optional),
    consumer(Common) }.
```

Entire words We can put the pieces together now by first defining the **`word`** constraint as the conjunction of syllabification and related prosodic constraints plus a classification of the word's segmental positions into **`initial,medial,final`** ones. Again, this is modulo interspersed *repeat* or *skip* symbols. This actually means that base syllabification and word syllabification must match up, but fortunately this is indeed a property of our Temiar data.

Second, **`wordform`** conjoins the previous constraint with its parameter **`X`** – which will contain the conjunction of stem and aspect morphemes –, before eliminating leftover consumer symbols with the help of **`closed_interpretation`**:

```
word := ignore(syllabification &
               prosodic_constraints &
               positional_classification,
               technical_symbols).

positional_classification :=
[consumer(initial),consumer(medial)*,
 consumer(final)].

wordform(X):=closed_interpretation(X&word).
```

These definitions have removed the last barrier to evaluating expressions like **`wordform(selog & simulfactive & causative)`** or even suitable disjunctive combinations of such expressions which define entire paradigms. Figure 3 shows an example automaton for three forms. We refrain from describing a final automaton operation called Bounded Local Optimization in (Walther, 1999b) that was put

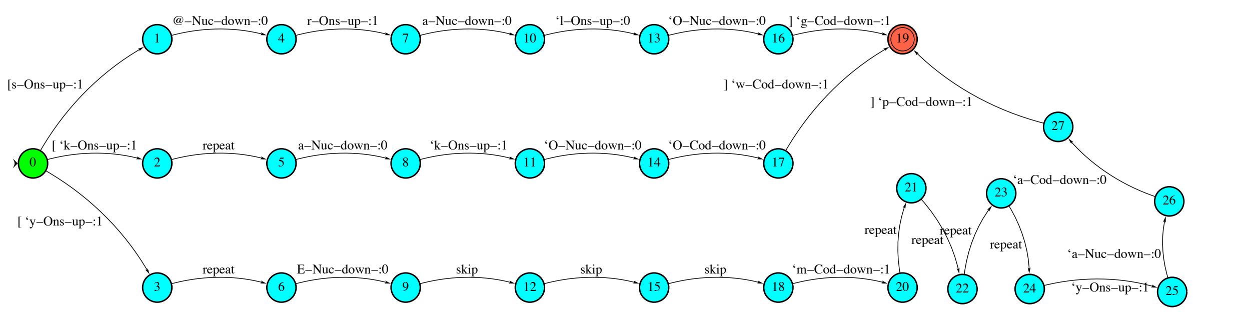

Figure 3: Temiar reduplications *səralɔg*, *kakɔɔw*, *yɛmyaap*

to use here to filter harmless spurious ambiguities from the original version of fig. 3. The kind of ambiguity involved in our Temiar grammar is one of alternative distribution of technical symbols in strings of the same segmental-content yield. Suffice to say that a simple parametrization of Bounded Local Optimization, which could only look at length-1 transition paths emerging from any given state, was able to prune the unwanted alternatives by considering technical transitions costlier in weight than segmental transitions.

5 Conclusion

The present paper has provided further support for (Walther, 1999b)'s finite-state conception of One-Level Prosodic Morphology by formulating – for the first time – a fully formalized and computational analysis of a complicated piece of reduplicative morphology found in the Mon-Khmer language of Temiar. Compared to the initial proposal, all three core components of enriched representations, namely technical transitions for repeating or skipping segmental symbols and the ability to perform infixation by using self loops, were again found necessary in the course of this analysis. However, in Temiar the last enrichment – *add_self_loops* – needed to be parametrized for a prosodic condition to narrow down the insertion site to a unique position per base.

The prosodic condition of ‘stressed onset’ proved crucial to define that position, and accounted for the variation between infixing aspectual morphology in longer bases and descriptively prefixing morphology in monosyllabic ones. Temiar thus underscores the utility of computing with real prosodic information in finite-state morphology, a frequently missing desideratum according to (Sproat, 1992, p.170). Also, the symmetry of having both forward and backward-pointing technical transitions in enriched automata representations was exploited in a novel regular expression operator called `seek(X)`, which encapsulated an interesting kind of ambiguous directional movement (or: movement underspecified for direction) towards a position satisfying property **X**. This operator could rather directly be motivated from the data. In particular, it facilitated an insightful account of the base-length-dependent triggering of reduplication in the active simulfactive aspect.

Finally, in contrast to even the most recent analyses in the theoretical linguistic literature, the full paradigm including the causative forms was captured in this fairly complete analysis, together with phonological modifications that sometimes occur between base and reduplicant, as exemplified by *yɛ***m***yaa***p**. Apart from an optional filtering step for some technical spurious ambiguities that could make use of local optimization, neither global optimization nor violable or soft constraints of the type argued for in Optimality Theory (Prince and Smolensky, 1993) were found necessary.

For future research, the empirical base of Temiar should be broadened to include further reduplication patterns, in particular those found in expressives. Also, the grammar should be amended to allow for words containing geminates, which were initially excluded to simplify the overall analysis at the cost of what is at best a peripheral aspect of it. Because the finite-state constraints employed in this work are all surface-true, the potential of machine-learning techniques to acquire them automatically from surface-oriented corpora should be explored. Finally, it would be very interesting to broaden to Temiar the ongoing experiments with efficiency-oriented computational variants of the One-Level Prosodic Morphology framework that were already alluded to in the text.

References

Evan Antworth. 1990. *PC-KIMMO: A Two-Level Processor for Morphological Analysis*. SIL, Dallas.

Kenneth R. Beesley. 1998. Constraining separated morphotactic dependencies in finite-state grammars. In *Proceedings of FSMNLP'98, Bilkent University, Turkey*, pages 118–127.

Geoffrey Benjamin. 1976. An outline of Temiar grammar. In Philip Jenner, Lawrence Thompson, and Stanley Starosta, editors, *Austroastiatic studies*, volume II, pages 129–187. University Press of Hawaii, Honululu.

Steven Bird and T. Mark Ellison. 1992. One-Level Phonology: Autosegmental representations and rules as finite-state automata. Technical report, Centre for Cognitive Science, University of Edinburgh. EUCCS/RP-51.

Steven Bird and T. Mark Ellison. 1994. One-Level Phonology. *Computational Linguistics*, 20(1):55–90.

Ellen Broselow and John McCarthy. 1983. A theory of infixing reduplication. *The Linguistic Review*, 3:25–98.

Adamantios Gafos. 1995. On the Proper Characterization of 'Nonconcatenative' Languages. Ms., Department of Cognitive Science, The Johns Hopkins University, Baltimore. (ROA-106 at http://ruccs.rutgers.edu/roa.html).

Diamandis Gafos. 1996. *The articulatory basis of locality in phonology*. Ph.D. thesis, The Johns Hopkins University, Baltimore, Md. [Published by Garland:New York].

Diamandis Gafos. 1998a. A-templatic reduplication. *Linguistic Inquiry*, 29(3):515–527.

Diamandis Gafos. 1998b. Eliminating long distance consonantal spreading. *Natural Language and Linguistic Theory*, 16(2):223–278.

Ron Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–78.

John McCarthy. 1982. Prosodic templates, morphemic templates, and morphemic tiers. In Harry van der Hulst and Norval Smith, editors, *The structure of phonological representations, part I*, pages 191–224. Foris, Dordrecht.

Alan Prince and Paul Smolensky. 1993. Optimality theory. constraint interaction in generative grammar. Technical Report RuCCS TR-2, Rutgers University Center for Cognitive Science.

Patricia Shaw. 1993. The prosodic constituency of minor syllables. In *Proceedings of the Eleventh West Coast Conference on Formal Linguistics*, pages 117–132, Stanford, CA. CSLI Publications. [Distributed by Cambridge University Press].

Kelly Sloan. 1988. Bare-consonant reduplication. In *Proceedings of the Seventh West Coast Conference on Formal Linguistics*, pages 319–330, Stanford, CA. CSLI Publications. [Distributed by Cambridge University Press].

Richard Sproat. 1992. *Morphology and Computation*. MIT Press, Cambridge, Mass.

Gertjan van Noord. 1997. FSA Utilities: A toolbox to manipulate finite-state automata. In Darrell Raymond, Derrick Wood, and Sheng Yu, editors, *Automata Implementation*, volume 1260 of *Lecture Notes in Computer Science*, pages 87–108. Springer Verlag. (Software under `http://grid.let.rug.nl/~vannoord/Fsa/`).

Markus Walther. 1999a. *Deklarative prosodische Morphologie: constraint-basierte Analysen und Computermodelle zum Finnischen und Tigrinya*. Niemeyer, Tübingen.

Markus Walther. 1999b. One-Level Prosodic Morphology. Marburger Arbeiten zur Linguistik 1, University of Marburg. 64 pp. (http://xxx.lanl.gov/abs/cs.CL/9911011).

Markus Walther. 2000. Finite-state Reduplication in One-Level Prosodic Morphology. In *Proceedings of NAACL-2000*, pages 296–302, Seattle/WA. North American Association for Computational Linguistics, Morgan Kaufman. (http://xxx.lanl.gov/abs/cs.CL/0005025).