Finite-State 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

Reduplication, a central instance of prosodic morphology, is particularly challenging for state-of-the-art computational morphology, since it involves copying of some part of a phonological string. In this paper I advocate a finite-state method that combines enriched lexical representations via intersection to implement the copying. The proposal includes a resource-conscious variant of automata and can benefit from the existence of lazy algorithms. Finally, the implementation of a complex case from Koasati is presented.

1 Introduction

In the past two decades computational morphology has been quite successful in dealing with the challenges posed by natural language word patterns. Using finite-state methods, it has been possible to describe both word formation and the concomitant phonological modifications in many languages, ranging from straightforward concatenative combination (Koskenniemi, 1983) over Semitic-style non-concatenative intercalation (Beesley (1996), Kiraz (1994)) to circumfixional long-distance dependencies (Beesley, 1998).

However, Sproat (1992) observes that, despite the existence of "working systems that are capable of doing a great deal of morphological analysis", "there are still outstanding problems and areas which have not received much serious attention" (ibid., 123). Problem areas in his view include subtractive morphology, infixation, the proper inclusion of prosodic structure and, in particular, reduplication: "From a computational point of view, one point cannot be overstressed: the copying required in reduplication places reduplication in a class apart from all other morphology." (ibid., 60). Productive reduplication is so troublesome for a formal account based on regular languages (or regular relations) because unbounded total instances like Indonesian noun plural (*orang-orang* 'men') are isomorphic to the copy language ww, which is context-sensitive.

In the rest of this paper I will lay out a proposal for handling reduplication with finite-state methods. As a starting point, I adopt Bird & Ellison (1994)'s One-Level Phonology, a monostratal constraint-based framework where phonological representations, morphemes and generalizations are all finite-state automata (FSAs) and constraint combination is accomplished via automata intersection. While it is possible to transfer much of the present proposal to the transducer-based setting that is often preferred nowadays, the monostratal approach still offers an attractive alternative due to its easy blend with monostratal grammars such as HPSG and the good prospects for machine learning of its surface-true constraints (Ellison (1992), Belz (1998)).

After a brief survey of important kinds of reduplication in §2, section §3 explains the necessary extensions of One-Level Phonology to deal with the challenges presented by reduplication, within the larger domain of prosodic morphology in general. A worked-out example from Koasati in §4 illustrates the interplay of the various components in an implemented analysis, before some conclusions are drawn in section §5.

2 Reduplication

A well-known case from the context-sensitivity debate of the eighties is the N-o-N reduplicative construction from Bambara (Northwestern Mande, (Culy, 1985)):

(1) a. wulu-o-**wulu** 'whichever dog'
b. wulunyinina-o-**wulunyinina** 'whichever dog searcher'
c. wulunyininafilèla-o-**wulunyininafilèla** 'whoever watches dog searchers'

Beyond total copying, (1) also illustrates the possibility of so-called fixed-melody parts in reduplication: a constant /o/ intervenes between base (i.e. original) and reduplicant (i.e. copied part, in bold print).[1]

The next case from Semai expressive minor reduplication (Mon-Khmer, Hendricks (1998)) highlights the possibility of an interaction between reduplication and internal truncation:

(2)	a.	cʔɛːt	**ct**-cʔɛːt	'sweet'
	b.	dŋɔh	**dh**-dŋɔh	'appearance of nodding constantly'
	c.	cfaːl	**cl**-cfaːl	'appearance of flickering red object'

Reduplication copies the initial and final segment of the base, skipping all of its interior segments, which may be of arbitrary length.

A final case comes from Koasati punctual-aspect reduplication (Muscogean, (Kimball, 1988)):

(3)	a.	ta.hás.pin	$\text{t}_1$ahas-$\mathbf{t}_1$óː-pin 'to be light in weight'
	b.	la.pát.kin	$\text{l}_1$apat-$\mathbf{l}_1$óː-kin 'to be narrow
	c.	ak.lát.lin	a_1k-$\mathbf{h}_1$o-látlin 'to be loose'
	d.	ok.cák.kon	o_1k-$\mathbf{h}_1$o-cákkon 'to be green or blue'

Koasati is particularly interesting, because it shows that copy and original need not always be adjacent – here the reduplicant is infixed into its own base – and also because it illustrates that the copy may be phonologically modified: the /h/ in the copied part of (3).c,d is best analysed as a voiceless vowel, i.e. the phonetically closest *consonantal* expression of its source. Moreover, the locus of the infixed reduplicant is predictable on prosodic grounds, as it is inserted after the first heavy syllable of the base. Heavy syllables in Koasati are long (C)VV or closed (C)VC. Prosodic influence is also responsible for the length alternation of its fixed-melody part /o(o)/, since the heaviness requirement for the penultimate, stressed, syllable of the word causes long [oː] iff the reduplicant constitutes that syllable.

[1]Culy (1985), who presents a superset of the data under (1) in the context of a formal proof of context-sensitivity, shows that the reduplicative construction in fact can copy the outcome of a *recursive* agentive construction, thereby becoming truly unbounded. He emphasizes the fact that it is "very productive, with few, if any restrictions on the choice of the noun" (p.346).

3 Finite-State Methods

The present proposal differs from the state-labelled automata employed in One-Level Phonology by returning to conventional arc-labelled ones, but shares the idea that labels denote *sets*, which is advantageous for compact automata.

3.1 Enriched Representations

As motivated in §2, an appropriate automaton representation of morphemes that may undergo reduplication should provide generic support for three key operations: (i) copying or repetition of symbols, (ii) truncation or skipping, and (iii) infixation.

For *copying*, the idea is to enrich the FSA representing a morpheme by encoding stepwise repetition locally. For every content arc $i \xrightarrow{c} j$ we add a reverse **repeat arc** $j \xrightarrow{repeat} i$. Following repeat arcs, we can now move backwards within a string, as we shall see in more detail below.

For *truncation*, a similar local encoding is available: For every content arc $i \xrightarrow{c} j$, add another **skip arc** $i \xrightarrow{skip} j$. This allows us to move forward while suppressing the spellout of c.

A generic recipe for *infixation* ensures that segmental material can be inserted anywhere within an existing morpheme FSA. A possible representational enrichment therefore adds a **self loop** $i \xrightarrow{\Sigma} i$ labelled with the symbol alphabet Σ to every state i of the FSA.[2]

Each of the three enrichments presupposes an epsilon-free automaton in order to be wellbehaved. This requirement in particular ensures that technical arcs (*skip*, *repeat*) are in 1:1 correspondence with content arcs, which is essential for unambiguous positional movement: e.g. $add\_skips(a\,\epsilon\,b)$ would ambiguously require 1 *or* 2 skips to supress the spellout of b, because it creates a disjunction of the empty string ϵ with *skip*. It is perhaps worth emphasizing that there is no special interpretation whatsoever for these technical arcs: the standard automaton semantics is unaffected. As a consequence, *skip* and *repeat* will be a visible part of the output in word form generation and must be allowed in the input for parsing as well.

Taken together, the three enrichments yield an automaton for Bambara *wulu*, shown in figure 1.a. While skipping is not necessary for this example, $4 \xrightarrow{\Sigma} 4$ is: it will host the fixed-melody /o/. The

[2]This can be seen as an application of the *ignore* operator of Kaplan and Kay (1994), where Σ^* is being ignored.

repeat arcs will of course facilitate copying, as we shall see in a moment.

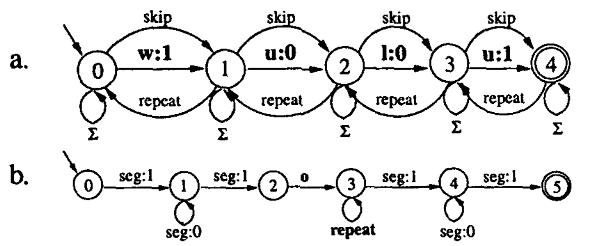

Figure 1: Enriched automata for *wulu* (a.), Bambara N-o-N reduplication (b.)

3.2 Copying as Intersection

Bird & Ellison (1992) came close to discovering a useful device for reduplication when they noted that automaton intersection has at least indexed-grammar power (ibid., p.48). They demonstrated their claim by showing that odd-length strings of indefinite length like the one described by the regular expression $(a\,b\,c\,d\,e\,f\,g)^+$ can be repeated by intersecting them with an automaton accepting only strings of even length: the result is $(a\,b\,c\,d\,e\,f\,g\,a\,b\,c\,d\,e\,f\,g)^+$.

Generalizing from their artifical example, let us first make one additional minor enrichment by tagging the edges of the reduplicative portion of a base with **synchronization bits :1**, while using the opposite value **:0** for the interior part (see figure 1.a). This gives us a segment-independent handle on those edges and a regular expression $seg_{:1}seg_{:0}{}^*seg_{:1}$ for the whole synchronized portion (*seg* abbreviates the set of phonological segments).

Assuming repeat-enriched bases, a total reduplication morpheme can now be seen as a partial word specification which mentions two synchronized portions separated by an arbitrary-length move backwards:

(4) $seg_{:1}seg_{:0}{}^*seg_{:1}$ ***repeat**** $seg_{:1}seg_{:0}{}^*seg_{:1}$

Moreover, total **reduplicative copying** now simply **is intersection** of the base and (4), or – in the Bambara case – a simple variant that adds the /o/ (figure 1.b). Disregarding self loops for the moment, the reader may verify that no expansion of the kleene-starred *repeat* that traverses less than $|base|$ segments will satisfy the demand for *two* synchronized portions. Semai requires another slight variant of (4) which *skips* the interior of the base in the reduplicant:

(5) $seg_{:1}$ ***skip**** $seg_{:1}$ *repeat** $seg_{:1}seg_{:0}{}^*seg_{:1}$

The identification of copying with intersection not only allows for great flexibility in describing the full range of actual reduplicative constructions with regular expressions, it also *reuses* the central operation for constraint combination that is independently required for one-level morphology and phonology. Any improvement in efficient implementation of intersection therefore has immediate benefits for grammar computation as a whole. In contrast, a hypothetical setup where a dedicated total copy device is sandwiched between finite-state transducers seems much less elegant and may require additional machinery to detect copies during parsing.

Note that it is in fact possible to compute reduplication-as-intersection over an entire lexicon of bases (see figure 3 for an example), provided that repeat arcs are added individually to each base. Enriched base FSAs can then be unioned together and undergo further automaton transformations such as determinization or minimization. This restriction is necessary because our finite-state method cannot express token identity as normally required in string repetition. Rather than identifying the same token, it addresses the same string position, using the weaker notion of type identity. Therefore, application of the method is only safe if strings are effectively isolated from one another, which is exactly what per-base enrichment achieves. See §3.4 for a suggestion on how to lift the restriction in practice.

3.3 Resource Consciousness

One pays a certain price for allowing general repetition and infixation: because of its self loops and technical arcs, the automaton of figure 1.a overgenerates wildly. Also, during intersection, self loops can absorb other morphemes in unexpected ways. A possible diagnosis of the underlying defect is that we need to distinguish between **producers and consumers of information**. In analogy to LFG's constraint vs constraining equations, information may only be consumed if it has been produced at least once.

For automata, let us spend a P/C bit per arc, with P/C=1 for producers and P/C=0 for consumer arcs. In *open interpretation* mode, then, intersection combines the P/C bits of compatible arcs via logical OR, making producers dominant. It follows that a resource may be multiply consumed, which has obvious advantages for our application, the multiple realization of string symbols. A final step of *closed in-*

terpretation prunes all consumer-only arcs that survived constraint interaction, in what may be seen as intersection with the universal producer language under logical-AND combination of P/C bits.

Using these resource-conscious notions, we can now model both the default absence of material and purely contextual requirements as consumer-type information: unless satisfied by lexical resources that have been explicitly produced, the corresponding arcs will not be part of the result. By convention, producers are displayed in bold. Thus, the exact result of figure 1.a $\cap$ 1.b after closed interpretation is:

$$w_{:1}\, u_{:0}\, l_{:0}\, u_{:0}\, \mathbf{o}\; \mathbf{repeat^4}\; \mathbf{repeat^*}\; w_{:1}\, u_{:0}\, l_{:0}\, u_{:1}$$

This expression also illustrates that, for parsing, strings like *wuluowulu* need to be consumer-self-loop-enriched via a small preprocessing step, because intersection with the grammar would otherwise fail due to unmentioned technical arcs such as *repeat*. Because our proposal is fully declarative, parsing then reduces to intersecting the enriched parse string with the grammar-and-lexicon automaton (whose construction will itself involve intersection) in closed interpretation mode, followed by a check for nonemptiness of the result. Whereas the original parse string was underspecified for morphological categories, the parse result for a realistic morphology system will, in addition to technical arcs, contain fully specified category arcs in some predefined linearization order, which can be efficiently retrieved if desired.

3.4 On-demand Algorithms

It is clear that the above method is particularly attractive if some of its operations can be performed online, since a fullform lexicon of productive reduplications is clearly undesirable e.g. for Bambara. I therefore consider briefly questions of efficient implementation of these operations.

Mohri et al. (1998) identify the existence of a *local* computation rule as the main precondition[3] for a *lazy* implementation of automaton operations, i.e. one where results are only computed when demanded by subsequent operations. Such implementations are very advantageous when large intermediate automata may be constructed but only a small part of them is visited for any particular input. They show that such a rule exists for composition $\circ$, hence also for our operation of intersection ($A \cap B \equiv range(identity(A) \circ identity(B))$).

Fortunately, the three enrichment steps all have local computation rules as well:

(6) a. $q_1 \xrightarrow{c} q_2 \;\Rightarrow\; q_2 \xrightarrow{repeat} q_1$

b. $q_1 \xrightarrow{c} q_2 \;\Rightarrow\; q_1 \xrightarrow{skip} q_2$

c. $q \;\Rightarrow\; q \xrightarrow{\Sigma} q$

The impact of the existence of lazy implementations for enrichment operations is twofold: we can (a) now maintain minimized base lexicons for storage efficiency and add enrichments lazily *to the currently pursued string hypothesis* only, possibly modulated by exception diacritics that control when enrichment should or should not happen.[4] And (b), laziness suffices to make the proposed reduplication method reasonably time-efficient, despite the larger number of online operations. Actual benchmarks from a pilot implementation are reported elsewhere (Walther, submitted).

4 A Worked Example

In this section I show how to implement the Koasati case from (3) using the FSA Utilities toolbox (van Noord, 1997). FSA Utilities is a Prolog-based finite-state toolkit and extendible regular expression compiler. It is freely available and encourages rapid prototyping.

Figure 2 displays the regular expression operators that will be used (italicized operators are modifications or extensions). The grammar will be presented below in a piecewise fashion, with line numbers added for easy reference.

`[]`	empty string
`[E1,E2, ... ,En]`	concatenation of $\mathtt{E}_i$
`{E1,E2, ... ,En}`	union of $\mathtt{E}_i$
`E*`	Kleene closure
`E^`	optionality
`E1 & E2`	intersection
X --> (Y/Z)	monotonic rule $X \rightarrow Y \subseteq X \;/\; \_\_ \; Z$
$\sim S$	complement set of S
Head(arg1, ... , argN) := Body	(parametrized) macro definition

Figure 2: Regular expression operators

[3]A second condition is that no state is visited that has not been discovered from the start state. It is easy to implement (6) so that this condition is fulfilled as well.

[4]See Walther (submitted) for further details. With deterministic automata, the question of how to recover from a wrong string hypothesis during parsing is not an issue.

Starting with the definition of stems (line 1), we add the three enrichments to the bare phonological string (2). However, the innermost producer-type string constructed by stringToAutomaton (3) is intersected with phonological constraints (5, 6) that need to see the string only, minus its enrichments. This is akin to lexical rule application.

```
1 stem(FirstSeg, String) :=
2  add_repeats(add_skips(add_self_loops(
3  [FirstSeg, stringToAutomaton(String)]
4    & ignore_technical_symbols_in(
5 moraification&mark_first_heavy_syllable
6    & positional_classification)))).
7
8 underspecified_for_voicing(BaseSpec) :=
9        { producer(BaseSpec & vowel),
10          [producer(h),consumer(skip)] }.
11
12 tahaspin   := stem([], "tahaspin").
13 aklatlin   := stem(underspecified_for_
14               voicing(low),"klatlin").
```

Lines 8-10 capture the V/h alternation that is characteristic for vowel-initial stems under reduplication, with the vocalic alternant constituting the default used in isolated pronunciation. In contrast, the /h/ alternant is concatenated with a consumer-type skip that requires a producer from elsewhere. Lines 12-14 define two example stems.

The following constraint (15-18) enriches a prosodically underspecified string with moras μ – abstract units of syllable weight (Hayes, 1995) –, a prerequisite to locating (20-24) and synchronization-marking (25-31) the first heavy syllable after which the reduplicative infix will be inserted.

```
15 moraification :=
16 ( vowel     --> ( mora / sigma ) )&
17 ( consonant --> ( mora / consonant ) )&
18 ( consonant --> ( (~ mora) / vowel ) ).
19
20 first_(X) := [not_contains(X), X].
21 heavy_rime := [consumer(mora),
22                consumer(mora)].
23 heavy_syllable := [consumer(~ mora),
24                    heavy_rime].
25 mark_first_heavy_syllable :=
26 [first_(heavy_rime)&synced_constituent,
27         synced_constituent].
28 right_synced := [consumer(~':1'&seg) *,
29                  consumer(':1'&seg)].
30 synced_constituent :=
31   [consumer(':1'&seg), right_synced].
32 positional_classification :=
33  [consumer(initial),consumer(medial) *,
34   consumer(final)].
```

Note that both the constituent before $(t_{:1}\ \overset{\mu}{a}\ h\ \overset{\mu}{a}\ \overset{\mu}{s}_{:1})$ and *after* $(p_{:1}\ \overset{\mu}{i}\ \overset{\mu}{n}_{:1})$ the infixation site need to be marked. Also, it turns out to be useful to classify base string positions for easy reference in the reduplicative morpheme, which motivates lines 32-34.

The main part now is the reduplicative morpheme itself (35), which looks like a mixture of Bambara and Semai: the spellout of the base is followed by iterated repeats (36) to move back to its synchronized initial position (37), which – recall /h/ – is required to be consonantal. The rest of the base is skipped before insertion of the fixed-melody part /o(o)/ occurs (38, 42-44). Proceeding with the interrupted realization of the base, we identify its beginning as a synchronized syllable onset (~ mora), followed by a right-synchronized string (39-40).

```
35 punctual_aspect_reduplication :=
36 [synced_constituent, producer(repeat)*,
37  consumer(':1' & initial & consonant),
38  producer(skip) *, fixed_melody,
39  consumer(':1' & seg & ~ mora),
40  right_synced].
41
42 fixed_melody :=
43 [producer(o & ~ ':1' & medial & mora),
44  producer(o & ~ ':1' & medial & mora)^].
```

Finally, some obvious word_level_constraints need to be defined (45-54), before the central intersection of Stem and punctual-aspect reduplication (57) completes our Koasati fragment:

```
45 word_level_constraints :=
46 last_segment_is_moraic &
47 last_two_sylls_are_heavy.
48
49 last_segment_is_moraic :=
50   [consumer(sigma) *, consumer(mora)].
51
52 last_two_sylls_are_heavy :=
53   [consumer(sigma) *,
54    heavy_syllable,heavy_syllable].
55
56 wordform(Stem):=closed_interpretation(
57   word_level_constraints & Stem &
58   punctual_aspect_reduplication).
```

The result of wordform({tahaspin,aklatlin}) is shown in figure 3 ([and] are aliases for initial and final position).

Space precludes the description of a final automaton operation called Bounded Local Optimization (Walther, 1999) that turns out to be useful here to

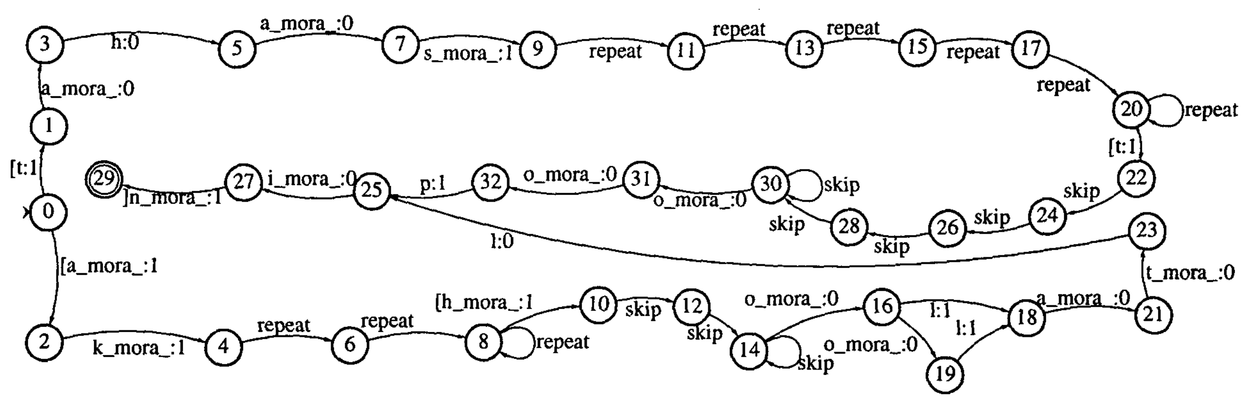

Figure 3: Koasati reduplications *tahas-too-pin*, *ak-ho(o)-latlin*

ban unattested free length variation, as found e.g. in *ak-ho(o)-latlin* where the length of *o* is yet to be determined. Suffice to say that a parametrization of Bounded Local Optimization would prune the moraic arc 16 $\rightarrow$ 19 in figure 3 by considering it costlier than the non-moraic arc 16 $\rightarrow$ 18, thereby eliminating the last source of indeterminacy.

5 Conclusion

This paper has presented a novel finite-state method for reduplication that is applicable for both unbounded total cases, truncated or otherwise phonologically modified types and infixing instances. The key ingredients of the proposal are suitably enriched automaton representations, the identification of reduplicative copying with automaton intersection and a resource-conscious interpretation that differentiates between two types of arc symbols, namely producers and consumers of information. After demonstrating the existence of efficient on-demand algorithms to reduplication's central operations, a case study from Koasati has shown that all of the above ingredients may be necessary in the analysis of a single complex piece of prosodic morphology.

It is worth mentioning that our method can be transferred into a two-level transducer setting without major difficulties (Walther, 1999, appendix B).

I conclude that the one-level approach to reduplicative prosodic morphology presents an attractive way of extending finite-state techniques to difficult phenomena that hitherto resisted elegant computational analyses.

Acknowledgements

The research in this paper has been funded by the German research agency DFG under grant WI 853/4-1. Particular thanks go to the anonymous reviewers for very useful comments.

References

Kenneth R. Beesley. 1996. Arabic finite-state morphological analysis and generation. In *Proceedings of COLING-96*, volume I, pages 89–94.

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

Anja Belz. 1998. Discovering phonotactic finite-state automata by genetic search. In *Proceedings of COLING-ACL '98*, volume II, pages 1472–74.

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.

Chris Culy. 1985. The complexity of the vocabulary of Bambara. *Linguistics and Philosophy*, 8:345–351.

T. Mark Ellison. 1992. *Machine Learning of Phonological Representations*. Ph.D. thesis, University of Western Australia, Perth.

Bruce Hayes. 1995. *Metrical stress theory: prin-*

ciples and case studies. University of Chicago Press.

Sean Hendricks. 1998. Reduplication without prosodic templates: A case from Semai. Handout from talk given at LSA annual meeting, January 8.

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

Geoffrey Kimball. 1988. Koasati reduplication. In W. Shipley, editor, *In honour of Mary Haas: from the Haas Festival Conference on Native American Linguistics*, pages 431–42. Mouton de Gruyter, Berlin.

George Anton Kiraz. 1994. Multi-tape two-level morphology: a case study in Semitic nonlinear morphology. In *Proceedings of COLING '94*, volume 1, pages 180–186.

Kimmo Koskenniemi. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production.* Ph.D. thesis, University of Helsinki.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 1998. A rational design for a weighted finite-state transducer library. In D. Wood and S. Yu, editors, *Automata Implementation. Second International Workshop on Implementing Automata, WIA '97*, volume 1436 of *Lecture Notes in Computer Science*, pages 144–58. Springer Verlag.

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. 1999. One-Level Prosodic Morphology. Marburger Arbeiten zur Linguistik 1, University of Marburg. 64 pp. (http://www.uni-marburg.de/linguistik/mal).

Markus Walther. submitted. On finite-state reduplication. In *COLING-2000.*