University of California

Los Angeles

Studies in Computational Optimality Theory, with Special Reference to the Phonological System of Malagasy

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Linguistics

by

Daniel Matthew Albro

2005

The dissertation of Daniel Matthew Albro is approved.

Bruce P. Hayes

Colin Wilson

D. Stott Parker

Edward P. Stabler, Committee Chair

University of California, Los Angeles

2005

Table of Contents

List of Figures

List of Tables

Vita

1971	Born, Cary, North Carolina, USA.
1991–1992	Programmer, C. S. Draper Labs.
1994	S.B. Computer Science and Computer Engineering, Massachusetts Institute of Technology.
1994	William A. Martin Memorial Thesis Prize for the top undergraduate thesis in Computer Science or Electrical Engineering.
1994–1995	Programmer/Analyst, CAS, Inc.
1995–2002	Research Assistant, Linguistics Department, UCLA.
1996, 1997	Title VI Fellowship for the study of Quechua.
1996–1997	Graduate Technology Consultant for the Linguistics Department, Humanities Computing Facility, UCLA.
1997, 1998	Instructional Developer, Latin American Center, UCLA.
1998–1999	Teaching Assistant, Linguistics Department, UCLA.
1998	M.A. Linguistics, University of California at Los Angeles.
1998–1999	Research Assistant, Learning and Information Systems project (joint project of the Linguistics, Biology, and Psychology Departments), UCLA.
1999	C.Phil. Linguistics, University of California at Los Angeles.
1999	Dissertation Year Fellowship.
1999–present	Senior Computational Linguist, Meaning Master Technologies.

Publications and Presentations

Albro, Daniel M. 1993. *AMAR: A Computational Model of Autosegmental Phonology*. MIT Artificial Intelligence Laboratory Technical Report 1450.

Albro, Daniel M. and Berwick, Robert C. 1995. *AMAR: A model of computational phonology*. Proceedings of the 2nd Holland Institute of Generative Linguistics Phonology Conference (HILP 2), Amsterdam.

Albro, Daniel M. 1998. *Three formal extensions to Primitive Optimality Theory*. Proceedings of the 17th International Conference on Computational Linguistics.

Albro, Daniel M. 2000. *A probabilistic ranking learner for phonotactics*. Proceedings of the 2000 conference of the Linguistics Society of America.

Albro, Daniel M. 2000. *Taking Primitive Optimality Theory beyond the finite state*. Proceedings of the 18th International Conference on Computational Linguistics, Special Interest Group in Computational Phonology.

Albro, Daniel M. 2003. *A Large-Scale, Computerized Phonological Analysis of Malagasy*. Proceedings of the 2003 meeting of the Linguistics Society of America.

Teal, Tracy; Albro, Daniel M.; Stabler, Edward P.; and Taylor, Charles. 1999. *Compression and Adaption*. In D. Floreano, J. Nicoud and F. Mondada, editors, Advances of Artificial Life (Proceedings of the 5th European Conference on Artificial Life), number 1674 in Lecture Notes in Artificial Intelligence, pages 709-719. Springer.

Abstract of the Dissertation

Studies in Computational Optimality Theory, with Special Reference to the Phonological System of Malagasy

by

Daniel Matthew Albro

Doctor of Philosophy in Linguistics

University of California, Los Angeles, 2005

Professor Edward P. Stabler, Chair

This dissertation examines Base-Reduplicant Correspondence Theory (BRCT) and, to a lesser extent, various approaches to phonological opacity via a two-pronged approach.

The first half of this dissertation develops a computational model of Optimality Theory which is capable of encoding BRCT analyses. The model may be taken as an extension or reworking of the weighted finite state models of Ellison 1994b; Eisner 1997c; Albro 1998a, in which the candidate set of Optimality Theory is modeled as a finite state machine and the constraints as weighted finite state machines. Whereas the constraint component of Optimality Theory is still modeled by finite state methods, the candidate set induced by GEN is represented by a higher member of the Chomsky hierarchy, Multiple Context Free Grammars (Seki *et al.* 1991). Also presented is a simple but complete representation of candidates and constraints that allows for an order-of-magnitude increase in efficiency over earlier representations.

The second half attempts to validate the model, presenting the full-scale phonological analysis of the Merina dialect of Malagasy, the primary language of Madagascar. Malagasy has an extensive system of reduplication and a highly opaque stress system. The analysis shows that Base-Reduplicant Correspondence Theory does not work as a model of Malagasy reduplication. Instead, a combination of Kiparsky's (2000) LPM-OT framework and a morphological doubling account similar to that of Inkelas & Zoll (2000) accounts for the patterns of Malagasy reduplication and opaque stress without requiring any computational power beyond the finite state.

CHAPTER 1

Introduction

This dissertation is part of a research program that attempts to provide a complete, implemented formal model of Phonological Optimality Theory (Prince & Smolensky 1993), where "complete" indicates that the model may be used to provide complete phonological analyses of the world's languages.

1.1 CONTEXT: PROS AND CONS OF OPTIMALITY THEORY

Optimality Theory is a grammar framework with many advantages for phonological analysis. Compared with analyses in previous rule-based frameworks, analyses in Optimality Theory arguably have more explanatory power, greater universality of application, and better coverage of many areas of the data. However, Optimality Theory has certain disadvantages with respect to the rule-based frameworks, and a large set of outstanding problems.

1.1.1 ADVANTAGES OF OPTIMALITY THEORY

Previous rule-based frameworks failed to integrate particular sound changes with the overall phonotactic pattern of the language (*cf.* Kisseberth (1970), who first pointed out this

problem). Optimality Theory allows the analyst to concentrate on characterizing generalizations that are true or mostly true of the *surface* form actually pronounced. The surface form is an appropriate form to concentrate on, as it is the level of phonological representation for which we have the most evidence. Further, within Optimality Theory there are many phonological phenomena that receive a comprehensive analysis for the first time, such as under-application and overapplication of reduplication, and "emergence of the unmarked" effects.

1.1.2 CURRENT DISADVANTAGES OF OPTIMALITY THEORY

However, Optimality Theory has a few disadvantages when compared with other frameworks.

In most phonological analyses of the sort where Occam's razor is respected, the underlying form of an utterance and its surface form tend to be quite similar, differing primarily to the extent that morphological juxtaposition of otherwise surface-possible forms creates violations of surface well-formedness conditions. In a rule-based analysis the reason for this is quite clear and built into the system itself—if no rule changes some aspect of the underlying form, then that aspect remains unchanged on the surface. This means that rule-based analysis consists essentially of noticing sound changes induced by the presence of a form in differing environments due to morphology and sentential context, and proposing a phonological rule for each type of change that takes place in a particular environment. Since this analytical process is reasonably straightforward and its correctness is fairly easy to check—rule-based generation can easily be simulated by hand—it was

possible for phonologists to create complex systems of phonological rules that covered all or most of the sound changes present in the vocabulary of a human language.

In Optimality Theory, on the other hand, the analyst must propose constraints and orderings to account not only for the ways in which surface forms differ from their underlying counterparts, but also for the ways in which they do *not* differ from them[1]. Furthermore, once an analysis is complete, its correctness is usually still far from obvious. To justify a particular constraint ranking, the analyst must show that the ranking proposed is best satisfied by the surface forms naturally found, and that none of the infinite set of potential outputs would better satisfy it. This could be done by simulating generation on the data set to be checked, but Optimality Theoretic generation is too complex to be performed by hand. As it is, justification of a constraint ranking requires the analyst to characterize a potentially infinite set of candidates and argue that all parts of the set are less well-formed according to the constraint ranking than the actual output.

Typically a ranking $\mathsf{A} \gg \mathsf{B}$ is justified by showing that the opposite ranking $\mathsf{B} \gg \mathsf{A}$ would favor a different output candidate. Such a ranking argument is insufficient, however—the analyst must also ensure that no constraint C where $\mathsf{C} \gg \mathsf{A}, \mathsf{B}$ exists that would independently rule out the candidate favored by the ranking $\mathsf{B} \gg \mathsf{A}$. In the vast majority of Optimality Theoretic analyses only a small sub-part of the phonology of a language is considered, so it is impossible to ensure that the entire constraint set for the language has been considered. Of course if the entire phonology of a language is considered, it becomes extremely difficult to check the large number of constraints that will probably be necessary to account for the data. As a result, since the advent of

[1] Rule-based accounts of phonotactics tend to have this problem as well.

Optimality Theory there has been a dearth of large-scale analyses in the vein of *The Sound Pattern of English* (Chomsky & Halle 1968) and *Spanish Phonology* (Harris 1969).

Because of the difficulty of justifying a particular analysis, several computational linguists have proposed formal models by which the computation of an optimal surface form from an underlying form might be carried out: Ellison 1994b and other methods using weighted finite state machines, Tesar 1996 and other methods involving context free grammars, and Karttunen 1998 and other methods involving finite state transducers. An analysis that can be shoehorned into one of these models can then be justified by showing that the model generates the correct output forms from the proposed input forms. However, none of these models are sufficiently developed (or computationally adequate) to encode the majority of actual analyses, due to lack of support for reduplication, long-distance metathesis, and opacity.

The problems discussed above make it difficult to analyze a large body of data and phonological phenomena in a consistent and demonstrably correct way. The work described here uses computational methods to get around the difficulties posed by OT and renders feasible a return to large-scale, whole-language analysis.

1.2 PRESENT CONTRIBUTION

Previous computational models of Phonological Optimality Theory have not accounted for reduplication. The mainstream approach to reduplication — Base-Reduplicant Correspondence Theory (McCarthy & Prince 1995) — appears, as will be seen, to rely upon computational power beyond the finite state. The same can be said for many approaches

to phonological opacity. This dissertation examines Base-Reduplicant Correspondence Theory (BRCT) and, to a lesser extent, various approaches to phonological opacity via a two-pronged approach.

The first half of this dissertation proposes a computational model of Optimality Theory which is capable of encoding BRCT analyses. The model may be taken as an extension or reworking of the weighted finite state models of Ellison 1994b; Eisner 1997c; Albro 1998a, in which the candidate set of Optimality Theory is modeled as a finite state machine and the constraints as weighted finite state machines. The principal innovation of my system is that whereas the constraint component of Optimality Theory is still modeled by finite state methods, the candidate set induced by GEN is represented by a higher member of the Chomsky hierarchy, Multiple Context Free Grammars (Seki *et al.* 1991). Also presented is a simple but complete representation of candidates and constraints that allows for an order-of-magnitude increase in efficiency over earlier representations.

The second half attempts to validate the model, presenting the full-scale phonological analysis of a language featuring problematic elements. The language chosen here was the Merina dialect (the standard dialect) of Malagasy, the primary language of Madagascar. Malagasy has an extensive system of reduplication and a highly opaque stress system. The analysis shows that Base-Reduplicant Correspondence Theory does not work as a model of Malagasy reduplication. Instead, a combination of Kiparsky's (2000) LPM-OT framework and a morphological doubling account similar to that of Inkelas & Zoll (2000) accounts for the patterns of Malagasy reduplication and opaque stress without requiring any computational power beyond the finite state.

CHAPTER 2

Optimality Theory with Multiple Context-Free Grammars

2.1 INTRODUCTION

As noted in the previous chapter, the grammar framework Optimality Theory (Prince & Smolensky 1993) has arguably increased both the explanatory force of linguistic analyses and the universe of phenomena amenable to analysis. However, it achieves this at the price of increased complexity. This increased complexity is multi-faceted, including complexity of conceptualization of the framework itself, complexity of analysis (including increased difficulty in proving the correctness of analyses), and complexity of computation. Chapter 3 touches upon complexity of analysis; here I look into other forms of complexity.

In order to get a clear conception of a grammatical framework, it is necessary to construct a formal model of it. There is such a model in Prince & Smolensky (1993)—this model will be reviewed in §A—but the model is not computable[1]. Further, the model given there is inadequate to account for various natural language phenomena, in particular, opacity, that is, cases where morphological considerations cause a surface form not to

[1] To be clear, the model as described there, with no limitations on the descriptive power of constraints, is not computable. This chapter is among a number of attempts to specify a computational model for Optimality Theory that is as close as possible to the description given in Prince & Smolensky (1993) while still being (feasibly) computable.

appear to be the most well-formed candidate for its underlying form, as pointed out by Steriade (1996), Kenstowicz (1995), McCarthy (1995), and Burzio (1995). Since the model is not computable as given, it appears unrealistic and therefore limits full conceptualization of the framework, precluding analysis of computational complexity and computer simulation (for the benefits of which, see Chapter 3). There have been a number of notable attempts to supply a computable model of phonological Optimality Theory. These include CFG parsing and dynamic programming (Tesar 1995), transducer methods (Karttunen 1998; Frank & Satta 1998; Eisner 2000), and the weighted finite state model (Ellison 1994b; Eisner 1997c; Albro 1998a). Unfortunately, none of these models has been shown to be capable of handling the full universe of phenomena explored in phonological Optimality Theoretic analyses. One of the purposes of this dissertation is to expand the Weighted Finite State model sufficiently to allow it to cover these phenomena. This chapter covers the phenomenon of reduplication (here defined as possibly imperfect copying of all or part of a word in order to fulfill a morphological function). Chapter 3 explores reduplication further, in the context of a large-scale analysis of Malagasy. It also takes on opacity and metathesis.

The most prevalent framework for analysis of reduplication within Optimality Theory is Base-Reduplicant Correspondence Theory (McCarthy & Prince 1995), which states that part of a reduplicating form is marked as a reduplicant, another part as a base of reduplication, and that constraints encourage similarity between the base and the reduplicant. This framework has been criticized on empirical grounds, for example by Inkelas & Zoll (2000) and will be further criticized in Chapter 3, below. It has also been criticized on formal grounds by Walther (2001), who provides an alternative model (Walther 2000),

based in the One-Level Phonology framework (Bird & Ellison 1994) rather than Optimality Theory. Earlier mention of reduplication as an area with higher computational complexity than other areas of phonology was made by Sproat (1992).

The remainder of this chapter reviews the weighted finite state model of Optimality Theory and shows that Base-Reduplicant Correspondence Theory may in fact be modeled fairly directly in an extended version of the model, and that the extended version is in fact computable, albeit less efficient than the original. The matters discussed here belong both to the disciplines of linguistics and computer science, both fairly broad fields. In order to keep the material accessible to practitioners of both, I have postponed rigid mathematical detail and proof to the appendices, in favor of exemplification and explanation.

2.2 THE WEIGHTED FINITE STATE MODEL OF OT

Before showing how to extend the weighted finite state model of Optimality Theory to model the Base-Reduplicant Correspondence Theoretical approach to reduplication, it is necessary first to explain what the weighted finite state model is. Furthermore, in order to explain the weighted finite state model of Optimality Theory, it is first necessary to explain the basic model of Optimality Theory[2].

The framework of phonological Optimality Theory is defined in terms of three components: GEN, CON, and EVAL. GEN is a function that takes an underlying form, such as /aha/, and produces a potentially infinite set containing anything that could be a possible output form for that underlying form in any human language. It is frequently assumed

[2] A more formal exposition of the weighted finite state model is given in Appendix A

that each of these *output candidates* contains within it a representation of the underlying form. For example, one element of the set produced by the application of GEN to input /aha/ might be [a:a h:– a:a]. The pronounced form here is [aa], and the underlying form is /aha/, *i.e.,* the /h/ was deleted. A more pictorial representation for this candidate might be the following coïndexing representation, in which subscripts indicate segment correspondence:

underlying: a_1 h a_2

surface: a_1 a_2

The next component is CON. This is a set of *constraints.* A constraint is an abstract device that *harmonically orders* the candidates produced by GEN. That is, it tells how good or bad a particular candidate is with respect to the others. In the mainstream tradition, a constraint is a well-formedness measure on candidates (or, more accurately, an ill-formedness measure); each constraint assigns a number of *constraint violations* to each candidate. For example, the constraint MAXIO states "assess a violation for any underlying segment that does not have a surface counterpart," *i.e.,* MAXIO penalizes deletion (alternatively, it prefers candidates that preserve underlying segments). Thus, the candidate [a:a h:- a:a] would have one violation of MAXIO. The general idea is that CON is the universal set of all constraints available to a human being in forming their grammar. An actual language-particular grammar consists of a *strict ranking* of these constraints. In a strict ranking, each constraint is infinitely more important than the constraints it outranks; *i.e.,* if constraint A prefers candidate a to candidate b, no lower-ranked constraint B can cause candidate b to win over candidate a (of course there could be a hypothetical constraint C that outranks A and causes b to win anyway). All of this leads up to the final compo-

nent, Eval. Eval is the mechanism by means of which it is determined what output form should be uttered to represent a given underlying form. This works (abstractly) as follows:

1. Take the output of Gen applied to the underlying form as the pool of candidate outputs.
2. Start with the highest ranked constraint (call it the current constraint, for now).
3. Evaluate each of the candidates in the pool with respect to the current constraint.
4. Calculate the minimum number of violations any candidate in the pool has.
5. Remove all candidate outputs that have more than the minimum number of violations.
6. If there is only one candidate in the pool, it's the output.
7. Otherwise, the current constraint is now the next highest ranked constraint.
8. If there are no more constraints, the output consists of whatever candidates there are in the pool.
9. Otherwise, proceed back to step 3.

The immediate reaction most people have when seeing this definition for the first time is to ask how this could possibly be computed. In particular, step 3 requires a function to be applied to a potentially infinite set of objects! Even if that is left aside, it is still necessary to figure out how one is to represent the ever-shrinking infinite pool of candidates. The answer to the first of these questions comes fairly naturally from the

answer to the second. Each candidate is a linguistic utterance (in computational terms, a *string*). Thus, the question is how to represent an infinite set of utterances. Luckily there is a known answer to this question — a *grammar* is a finite representation of a potentially infinite set of utterances. The next question, then, is how to evaluate all of the strings described by a grammar (that is, the set of grammatical strings with respect to said grammar) in terms of a constraint and produce a grammar that contains exactly the (again, possibly infinite) set of optimal strings with respect to the constraint. Several potential answers have been offered. As previously mentioned, these include CFG parsing and dynamic programming (Tesar 1995), transducer methods (Karttunen 1998; Frank & Satta 1998; Eisner 2000), and the weighted finite state model (Ellison 1994b; Eisner 1997c; Albro 1998a). I will explore the Weighted Finite State approach. In this approach a candidate set is represented as a *finite state machine*. A finite state machine is a computationally convenient representation of a *Regular Grammar*, which is one of the simplest grammar formalisms capable of representing an infinite set of strings. This sort of grammar is a rewrite grammar with two types of productions: $A \rightarrow aB$ and $A \rightarrow \epsilon$. An example should make this clear:

$$
\begin{aligned}
S &\rightarrow aA \\
A &\rightarrow aA \\
A &\rightarrow bA \\
A &\rightarrow aB \\
B &\rightarrow \epsilon
\end{aligned}
$$

In this grammar, S is a special non-terminal symbol referred to as the *start symbol*. A rewrite

grammar describes the set of strings that can be produced by the following procedure (known as a *derivation*):

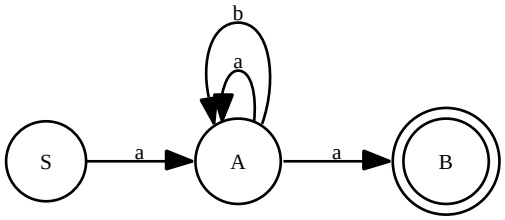
1. Let the *current string* be the start symbol S.
2. If the current string contains no non-terminal symbols (non-terminals are represented by uppercase letters; they are the only symbols that appear in the left-hand side of productions in a regular grammar), the current string is the final output of the procedure.
3. Otherwise, select an arbitrary non-terminal symbol from the current string. Replace this symbol with the right-hand side of a production headed by that symbol (if the right-hand side of the production is ϵ, which represents the empty string, delete the non-terminal symbol and replace it with nothing). Repeat step two.

For example, one derivation from the above grammar is the following:

1. S [*start symbol*]
2. aA [S → aA]
3. aaA [A → aA]
4. aaaB [A → aB]
5. aaa [B → ϵ]

Therefore, the string aaa is a member of the language (set of strings) denoted by the grammar.

It was mentioned above that candidate sets are not represented as regular grammars. Instead, they are represented via an equivalent formalism, finite state machines. A finite state machine is simply a representation of a regular grammar as a directed graph. Each non-terminal symbol in the grammar is a node in the graph, and each production of the form $A \rightarrow aB$ is an edge of the graph. The non-terminal symbols that head productions of the form $A \rightarrow \epsilon$ are noted as *final states*. Here is the example grammar as a finite state machine:



Using the finite state formalism, a string is a member of the language represented by a machine if it is possible to trace a path from the start state of the machine (usually labeled "1," but here labeled "S") to an accepting state where the path is made up of edges labeled by the elements of the string. For example, this machine accepts "aaa" because one can take the arc labeled "a" from state "S" to state "A," then the arc labeled "a" from state "A" back to state "A," and finally the arc labeled "a" from state "A" to state "B."

There is one more equivalent formalism that comes in handy: the *regular expression*. A regular expression is an expression of an algebra over the members of some alphabet, using the operations concatenation, union, and Kleene star. The most basic regular expression is made up of a single alphabetic symbol, *e.g.,* a. The result of the *concatenation* operation applied to two regular expressions is a regular expression denoting the set of strings where

the first part of each string comes from the first expression and the second from the second expression. For example, a concatenated with b is the expression ab (more examples to follow). The union of two regular expressions is a regular expression that denotes the union of the sets denoted by the operands. For example, a unioned with ab contains the strings "a" and "ab." This is written a|(ab). Finally Kleene star applied to a regular expression denotes the concatenation of the regular expression with itself zero or more times. For example, a^* denotes the set containing the empty string, "a," "aa," "aaa," etc. A regular expression equivalent to the example grammar presented above is $a(a|b)^*a$. That is, each string in the set begins with "a," followed by zero or more elements of the set {a,b}, and ending with "a." Another way of saying this is that it is the set of strings from the alphabet $\{a, b\}$ that begin and end with the symbol "a" and are at least two elements long.

There is a reason that one might want to use finite state machines: the set of finite state machines over a particular alphabet is closed under intersection, and intersection of finite-state machines is fairly efficient to compute. Thus, one might represent a binary constraint (a constraint that any given form either violates or does not violate) as a finite machine that accepts just those strings that do not violate the constraint. Then, intersecting that machine with the machine representing the pool of candidates will either indicate that the machines do not intersect, in which case the constraint will have no effect, or it will produce the set of candidates that obey the constraint. Intersection is a fairly simple process. It essentially works as follows:

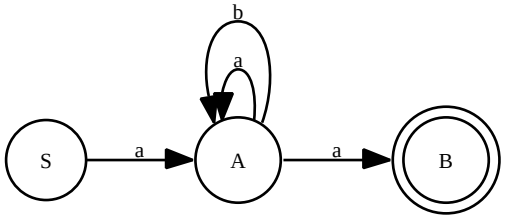
1. Create a new finite state machine whose states are labeled with pairs of states from the machines being intersected. The start state is labeled $\langle 1, 1\rangle$, indicating that it

corresponds to state 1 in both machines.

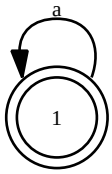
2. If symbol a labels an arc from state s_1 to state d_1 in the first machine, and it labels an arc from s_2 to state d_2 in the second machine, it labels an arc from $\langle s_1, s_2 \rangle$ to $\langle d_1, d_2 \rangle$in the intersection.

3. Any state $\langle s_1, s_2 \rangle$ in the result machine where s_1 is an acceptor in the first machine and s_2 is an acceptor in the second machine, is an acceptor.

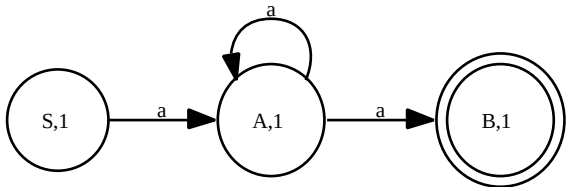
Here's an example. Assume that the candidate set is described by the machine given above, repeated here for convenience:



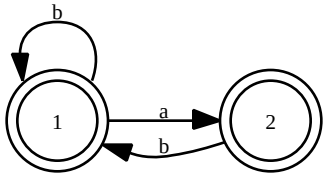
Then, assume that there is a constraint that bans the symbol b. As a binary constraint of the form described above, this would appear as follows:



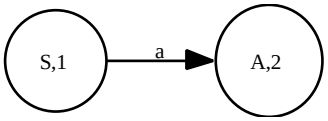
The intersection proceeds by first creating the product states $\langle S, 1\rangle$, $\langle A, 1\rangle$, and $\langle B, 1\rangle$. Of these, $\langle B, 1\rangle$ is a final state. Applying rule (2) above, we get the following intersection:



The a-labeled arcs all survived because there were a-arcs in both machines, but the b-labeled arc in the first machine had no counterpart. Another example might be a constraint that bans sequences of a's:

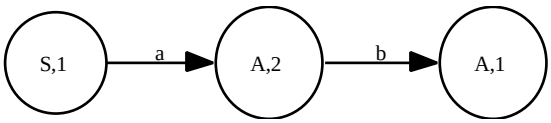


Here it is necessary to be more systematic. Let us begin at state S in the first machine and state 1 in the second. From here, a goes to state A (first machine) and state 2 (second machine), and b goes nowhere in the first machine, and to state 1 in the second. So far, therefore, the result machine appears as follows:

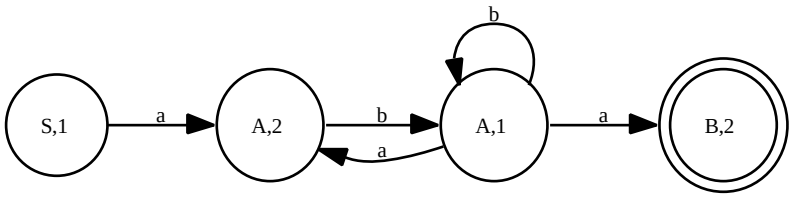


From state $\langle A, 2\rangle$, a goes to states A and B (first machine) and nowhere (second machine). Since it only appears in one machine, we do not add any arcs for a. The symbol b goes to

state A (first machine) and state 1 (second machine). Therefore we add a corresponding arc:



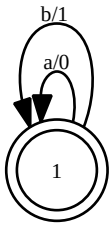
From state $\langle A, 1\rangle$, a goes to states A and B (first machine) and state 2 (second machine). Symbol b goes to state A (first machine) and state 1 (second machine). We add three arcs:



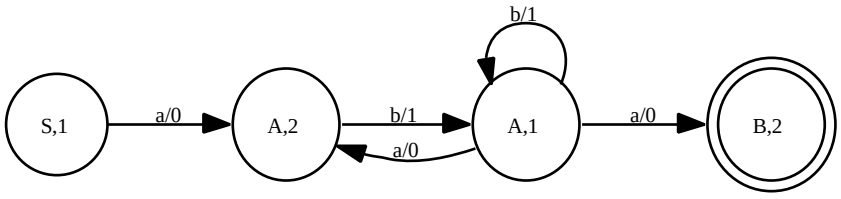
No arcs lead from state B in the first machine, so we are done. Notice that *this* pool of candidates does not include any that satisfy the "no b" constraint, so no intersection would be possible (following this algorithm you would get a machine with no final states that are reachable from the start state).

The major problem with the method laid out above is that not all constraints are binary. For example, one would usually expect a constraint like *B to incur one violation for a form with one b, two for two b's, and so forth. One solution to this is to employ *weighted* finite state machines. A weighted finite state constraint is a weighted finite state machine that accepts all potential candidates (every string that can be written in the given alphabet), but applies a weight to suboptimal candidates.

For example, *B would be represented as follows:

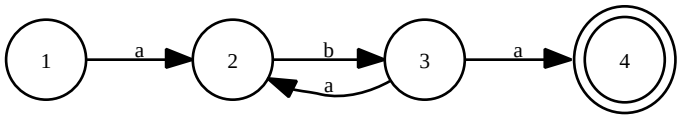


Note that every edge has a weight, that all strings from the alphabet {a, b} are accepted, and that the b edge has a non-zero weight. With a constraint like this, the number of constraint violations incurred by a given candidate can be computed by finding the lowest-weight path from the start state to a final state that follows edges labeled by the symbols in the candidate and summing the weights traversed. For example, "abb" here would go from state 1 to state 1, picking up 0 weight and consuming "a," then going from 1 to 1 again, picking up weight 1, and finally the same, for a total of weight 2. It is also straightforward to use one of these weighted constraints to evaluate an entire candidate pool at the same time, via weighted intersection. If we were to intersect this constraint with the candidate set given above, the result would be as follows:



Note that no path from the initial state to the final state will have a non-zero weight, but some paths are better than others. If we use Dijkstra's Single-Source Shortest Paths algorithm to figure out which paths to the final state are optimal, we find that the optimal

paths are those that do not include the edge from $\langle A, 1\rangle$ to $\langle A, 1\rangle$. Thus, the candidate set can be winnowed to the optimal one by removing the suboptimal arcs and dropping the weights:



More detail on this can be found in Albro (1998a), which notes that the shortest-paths algorithm as presented by Eisner (1997c) actually calculates the shortest paths to all final states, whereas what is desired is actually the shortest path to *any* final state. That is, if the candidate set FSM has multiple final states and all of the candidates that go to one of them are worse than all of the candidates that go to the other, applying the shortest paths algorithm without modification results in keeping suboptimal candidates. The solution is to combine the multiple final states into a single state before applying the shortest paths algorithm.

2.3 CONSTRAINTS AND REPRESENTATIONS

The previous section showed in general how to emulate the evaluation mechanism of Optimality Theory via weighted finite state methods. However, there is a large gap between having a computational model and actually being able to encode human-language analyses in it. The missing ingredients include a representation for candidates and an encoding of Optimality Theoretic constraints as finite state machines. The problem of representing phonological forms in a way that facilitates efficient computation on those forms has been

addressed many times.

The KIMMO project and its various incarnations (Koskenniemi 1983; Karttunen 1983) represent a phonological form as a string of symbols, each of which represents a phoneme or boundary (morphological, word, or syllable, generally). This method is quite efficient, but does not include the idea of incorporation where the representation contains both the underlying and surface forms. This makes the method unsuitable for a weighted finite state approach such as the one discussed here, but the transducer approach to Optimality Theory (Karttunen 1998; Frank & Satta 1998) typically uses essentially the same method to good effect (although with perhaps less coverage up to this point in the area of reduplication and long-distance metathesis).

Albro (1994) evaluates both the KIMMO representational scheme and the scheme introduced by the Delta Programming Language (Hertz 1990) with respect to their utility for representing forms with prosody and autosegmental tiers. It also includes its own representation, but that representation (a directed acyclic graph) is not particularly appropriate for finite state representation. The Delta Programming Language uses a metrical grid to represent phonological forms, but, not being designed for finite state representation, it is a bit too unwieldy for present purposes.

The most fully specified representation scheme for finite state phonology with prosody and autosegmental tiers (and incorporated underlying forms) is that of Primitive Optimality Theory (OTP) (Eisner 1997c; Eisner 1997d; Eisner 1997b). In Primitive Optimality Theory all representations take the form of a grid composed of the five characters '+', '-', '[', ']', and '|'. For example, a closed syllable with a sonorant in the nucleus and an obstru-

ent in the coda, followed by an open syllable headed by an obstruent might be partially represented as follows, where the underlying form is the same (the labels are not actually part of the representation)[3]:

```
syllable:  [ + + + + + | + + + ]
    mora:  - - [ + + + ] - [ + ]
syllabic:  - - [ + ] - - - [ + ]
sonorant:  [ + + + ] - - - [ + ]
                ⋮
```

Further tiers would be necessary to specify the other features and prosodic levels, and another set of tiers within the representation would be required to represent the underlying form of this utterance. For example, if the underlying form was the same, but syllable and mora structure was assigned by rule, this might look as follows:

```
syllable:  [ + + + + + | + + + ]
    mora:  - - [ + + + ] - [ + ]
syllabic:  - - [ + ] - - - [ + ]
sonorant:  [ + + + ] - - - [ + ]
                ⋮
syllabic:  - - [ + ] - - - [ + ]
sonorant:  [ + + + ] - - - [ + ]
```

[3] In general, I will put the actual parts of representations in non-italicized `monospace font`. Abbreviations and labels will be italicized.

Albro (1998a) argues that this representation has a number of flaws as a representation for Optimality Theory; in particular, there is no clear distinction between the absence of material at a level (as in the case where a segment has been deleted in the surface form but is present in the underlying form) and negative feature values. The amended version of the representation proposed by Albro (1998a) fixes these flaws at the expense of introducing one new tier per level of representation, a tier intended to indicate absense of material. Unfortunately, the presence of these tiers and the complication of having three symbols to indicate boundaries makes representation of reduplicated forms overly complicated. It also greatly increases the number of states necessary in constraints that evaluate forms in the representation.

There are two basic ways a Primitive Optimality Theoretic representation can be reduced to a string suitable for use with a finite state machine. Either each column of the grid can be taken as a single edge label, *i.e.*, as a single alphabet symbol, or the representation can be laid out column-by-column ("[–[...+–+...+[[+..." (row-by-row would be less feasible because whereas the number of columns in a given language might be fixed, the number of rows could not be, and, besides, such a representation would make constraints extremely difficult to formulate). The implementations of Eisner (1997c) and Albro (1998a) take the former tack.

The problem with the large-alphabet model, however, is that it has a very large alphabet, on the order of 5^n where n is the number of tiers. Many essential algorithms, such as Hopcroft's (1971) finite state machine minimization algorithm, have a time complexity based on the size of a finite state machine's alphabet, so a large alphabet can be a problem. The column-by-column small-alphabet scheme poses an even larger problem, however, as

the crucial Single-Source Shortest Paths algorithm takes time proportional to the (square of) the number of edges in the finite state machine (Dijkstra 1959), and the number of edges is based on the number of alphabetical symbols used in a given representation. The small-alphabet scheme does have one major advantage over the large-alphabet scheme, though—the same representation can be used for the finite state machine algorithms and also the parsing algorithms that will be necessary later to handle reduplicating forms.

The solution proposed here is essentially an amalgam of the small-alphabet and large-alphabet schemes for the Primitive Optimality Theoretic representation. It has the added bonus that it dispenses with the symbols '[', ']' and '|', replacing them by the equivalent of one additional tier. This means that the representations are significantly shorter. Further, the insertion/deletion problems referred to by Albro (1998a) are bypassed without additional complication. Basically, the idea starts with Eisner's representation and removes the edge symbols:

syllable: + + + + +
mora: - + + - +
syllabic: - + - - +
sonorant: + + - - +
⋮
syllabic: - + - - +
sonorant: + + - - +

Next, a partial version of the "large alphabet" system is applied, using a symbol to abbreviate a column of features. This is done in the conventional way, with phonetic symbols:

```
syllable:  s s s s s
    mora:  - m m - m
      SR:  n a t t a
      UR:  n a t t a
```

Note here that the representation above actually encodes more information than the Eisnerian one it replaces. It also has a problem: it is not possible to determine how many syllables there are on the syllable tier, how many t's there are in the SR and UR tiers, etc. The solution to this is to add a *continuation* and an *initial* version of each symbol. The continuation version of the symbol indicates that it represents the same entity as the previous symbol in its row. The initial version indicates that it is a different entity from the previous symbol in its row. Here, the continuation version is marked with a comma, and the initial version with a period ('-' is not marked with a comma or a period because it is a different sort of thing; see below):

```
syllable:  s. s, s, s. s,
    mora:  -  m. m. -  m.
      SR:  n. a. t. t. a.
      UR:  n. a. t. t. a.
```

This representation is now complete. It is possible here to determine that there are two syllables and three moras, and that there are five phones at each level of the representation.

Note that '-' indicates the lack of something at a given position, not the minus feature. Thus, if the surface form had an inserted final vowel and a deleted initial consonant, the form might appear as follows:

syllable: s. s, s, s. s,
mora: - m. m. - m.
SR: - a. t. t. a.
UR: n. a. t. t. -

This representation has no more ambiguity of the sort by which the Primitive Optimality Theoretic representation was plagued. The serialized form of this, suitable for encoding as a finite state machine, is "s. - - n. s, m. a. a. s, m. t. t. s. - t. t. s, m. a. -". Note that this is much shorter than the OTP representation would be if serialized in the same way (syllable, mora, surface phonetic, underlying phonetic).

For ease of constraint family specification, I have added one more wrinkle to the candidate specifications. At each end of each tier of an utterance, there must be a word-edge boundary symbol ('#'):

syllable: # s. s, s, s. s, #
mora: # - m. m. - m. #
SR: # - a. t. t. a. #
UR: # n. a. t. t. - #

2.3.1 REPRESENTATION OF CONSTRAINTS

Primitive Optimality Theory comes equipped with a very simple language for representing constraints, a language which necessarily depends upon its representation for candidates. Thus, if I discard the Primitive Optimality Theoretic representation in favor of the one proposed here, it is necessary to devise a new system of constraints. For this, I will introduce yet another compromise. This time, it is between Eisner's (1997d) idea that there are only two basic families of constraints, and each individual constraint is specifiable as a parameterization of one of these two and Ellison's (1994a) idea that any constraint encodable as a weighted finite state machine is permissible. What I have done is to take the families of constraints that are common in the Optimality Theoretic literature and develop weighted finite state templates for a useful subset of them. Like Eisner's, these constraints have in common that they are represented by weighted finite state machines that accept all strings, but apply weights to some strings. Further, no edge is weighted more heavily than 1. The families I have up to this point found useful are described in Chapter 3. An example of one such family follows.

2.3.2 CONSTRAINT FAMILY EXAMPLE: DEP-IO

Each family of constraints takes one or more parameters, each of which is a *natural class*, that is, a set of phonemes (to be a natural class, the set has to have something in common, such as a similarity in articulation). For example, DEPIO(S) is a family of constraints that penalizes instances where a member of natural class S appears in the surface representation, but no corresponding segment appears in the underlying form. Say that the total segment

inventory of a language is {a,b,h}, of which {a} constitutes the vowels (call it natural class *V*) and {b,h} constitutes the consonants (C). Take natural class C, the set of consonants. Then DepIO(C) might be represented by the weighted finite state machine in Figure 2.1.

This machine is fairly complicated, so I'll introduce some abbreviations:

- An edge labeled "C." abbreviates a set of edges, each edge of which is labeled by one of the initial consonants "b." and "h.".
- An edge labeled "C," abbreviates a set of edges labeled by the continuation consonants "b," and "h,".
- "C" indicates an edge set that is the union of "C." and "C,".
- "V." indicates an edge labeled "a.".
- "V," indicates an edge labeled "a,".
- "V" indicates the set of edges "a." and "a,".
- "X" indicates the set of edges that is the union of "C" and "V".
- "?" indicates the set of edges that contains every edge in "X", the edge labeled "-" (indicating the lack of a segment), and the edge labeled "#" (indicating the end of a word).

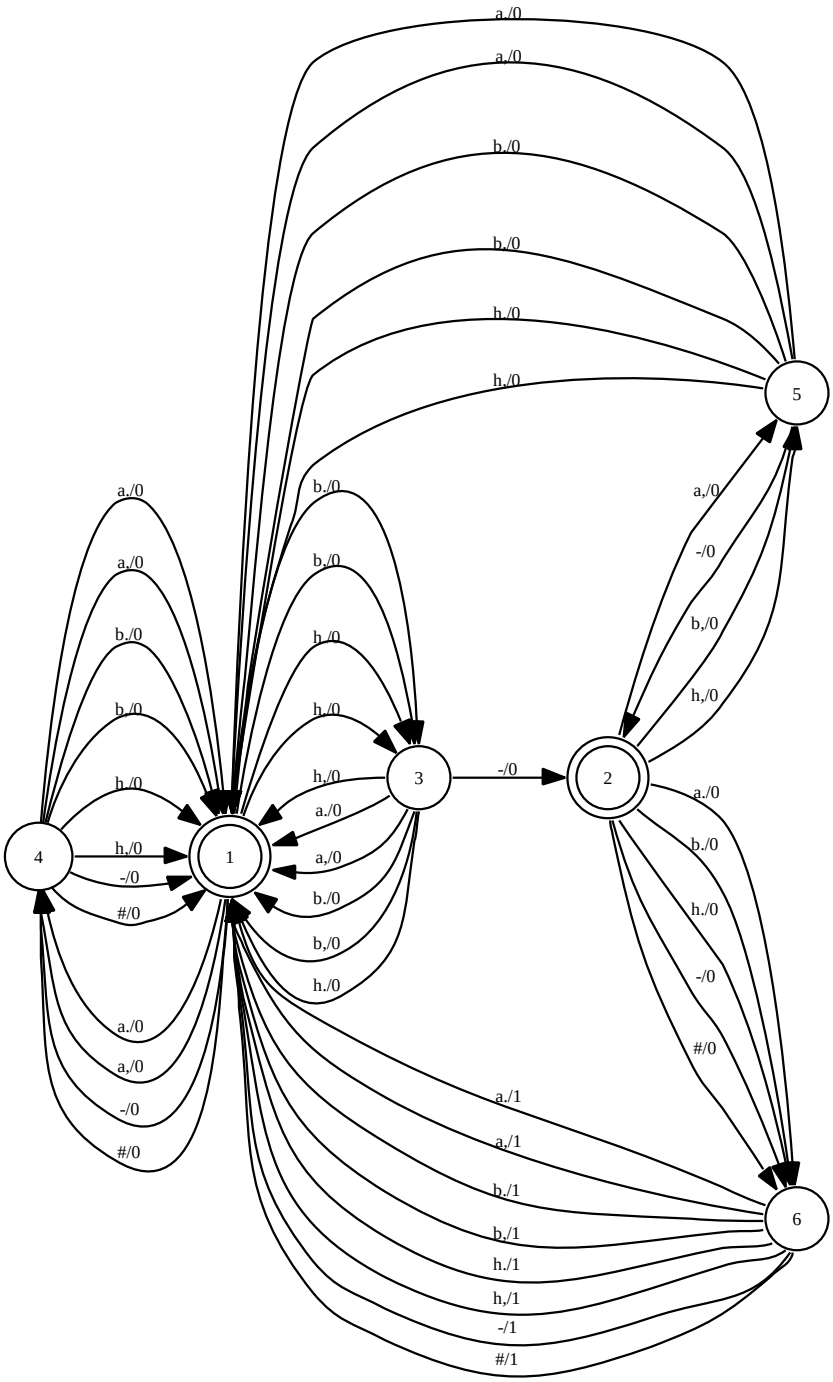
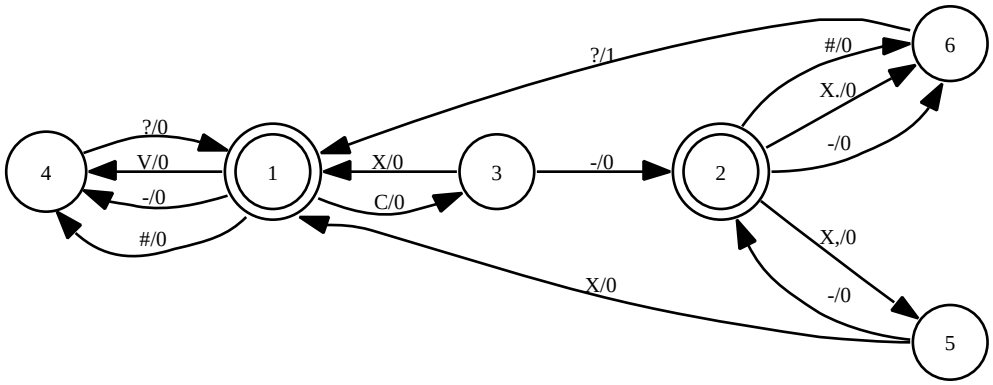


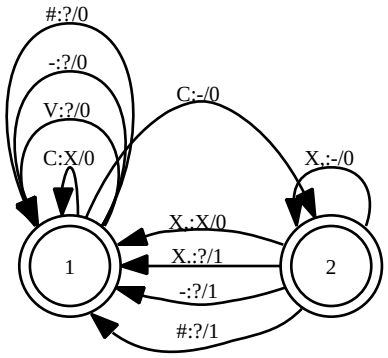
Figure 2.1: Unsimplified DepIO(C) representation

The abbreviated machine appears as follows:



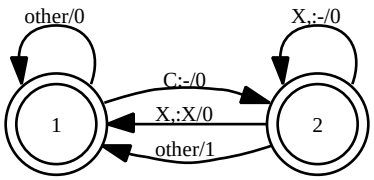
This machine is much easier to read. State 1 is the start state of the machine, and states 1 and 2 are the acceptors. Note that all paths through the machine from a start state to an acceptor are of a length that is a multiple of 2. This is because the representations evaluated by this constraint have two rows: SR and UR (the syllable and mora rows referred to above are not used here, nor in any of the examples to follow in this chapter). One further abbreviation makes the machine even easier to read. The abbreviation consists of putting the parts of an entire column of the representation together, separating the rows by means of the symbol ':'. For example, an edge labeled "X:-", and drawn from state s_1 to state s_2 indicates a set of edges leading from s_1 to some unrepresented state s_1' where each edge is labeled by one of the elements of "X" (see above), and an edge from s_1' to s_2 labeled by "-".

This method of abbreviation leads to the following machine:



Finally, one may notice from the machine above that some of the edges described are more meaningful than others. For example, the edge from state 1 to state 2 labeled "C:-/0" is fairly meaningful—it indicates that when a surface consonant is paired with nothing on the underlying form, there is a chance that DEPIO(C) will be violated (the only weighted edges originate in state 2). The other edges from state 1 simply indicate that in every other case rather than "C:-" the machine will stay in state 1, which could be called the "safe" state. This suggests a final abbreviation: if an edge from state s_1 to state s_2 is labeled "other", it indicates a set of edges labeled by all symbol pairs (in the case of representations with two rows) derivable from the alphabet of the language that are not covered by other edges from state s_1.

This should become clear from the example:



Now this machine is finally fairly simple to read. It may be interpreted as follows:

> Start in state 1 (the "safe" state). If a surface consonant is encountered, paired with nothing on the underlying form, move to state 2 (the "danger" state); otherwise, stay in state 1. From state 2, stay in state 2 if a surface continuation segment is encountered, still paired with nothing on the underlying form. On the other hand, if a surface continuation segment is paired with something on the underlying form, move to safety in state 1. If anything else is encountered, it signifies that the surface consonant had no underlying correspondent, and therefore one should pass through a weighted edge to state 1, where the process begins again.

It should be possible to see that this implements DepIO(C).

Now for something concrete: an example of how the Weighted Finite State system described above, using the representation and constraint families described above, can handle one of the basic Optimality Theoretic examples given in McCarthy & Prince 1995.

2.4 OPTIMALITY THEORETIC EXAMPLE: JAVANESE /H/-DELETION

This section will work through a slightly simplified version of [McCarthy and Prince's (1993, p5)] example of reduplicative over-application in Javanese. Take first a language, let's call it Pseudo-Javanese, with three phonemes: /a/, /b/, and /h/. For such a system of phonemes, only two features are needed: [±CONS] and [±SPREAD-GLOTTIS]. The phoneme /a/ is a vowel: [−CONS, −SPREAD-GLOTTIS]; /b/ is a non-spread-glottis consonant: [+CONS, −SPREAD-GLOTTIS]; and /h/ is a spread-glottis consonant: [+CONS, +SPREAD-GLOTTIS]. In this language, [h] may not appear between vowels, thus adding the suffix /-a/ leads to /h/-deletion. In reduplication this carries over so that both the reduplicant and the base exhibit /h/-deletion even though the conditioning context only occurs in one of them:

root	(a) isolation	(b) +a sfx	(c) redup'd	(d) red,+a
(1) /ah/	[ah]	[aa]	[aah]	[aaa]
(2) /bah/	[bah]	[baa]	[bahbah]	[babaa]

Let us begin with form (1b). The input to GEN is the underlying form, /aha/. For this example only two tiers will be necessary: the underlying representation (UR) and the surface representation (SR). Therefore, from the input "a h a", GEN produces a set of candidates with two tiers. These candidates include all forms in which /aha/ is the underlying form and the surface may be anything (so any insertion or deletion is possible):

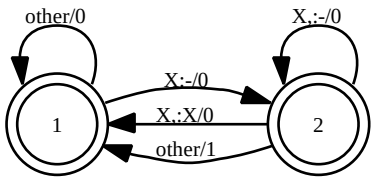
SR: # *X** *?* *?** *X** *?* *?** *X** *?* *?** *X** #

UR: # - a. a, - h. h, - a. a, - #

Note that the candidate set representation above is a special form of regular expression in which 'X' indicates that any member of the set {a,b,h} of permissible segments can appear; '?' indicates that any member of the set represented by 'X' may appear, as well as '-' (which indicates deletion); '*' indicates that the column it is in may appear 0 or more times; '.' indicates the first element of a given segment; ',' indicates the continuation of a segment; and '#' indicates the beginning or end of a word.

The candidate set embodied by the above regular expression, the output of GEN, will now be winnowed down by the constraints of the analysis (these are essentially the constraints given in McCarthy & Prince 1995, with some additional constraints needed in order to show a complete derivation), in order from the highest ranked to the lowest ranked.

DEPIO(X): The highest constraint in the ranking is DEPIO(X), where 'X' as usual represents the set of all segments (both the continuation and initial versions). The constraint indicates that no segment is to be inserted. Here is a representation of the constraint in the simplified scheme developed in §2.3:



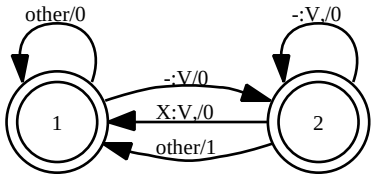
The candidate set that results from the intersection is as follows:

SR: # ? ?* ? ?* ? ?* #

UR: # a. a, h. h, a. a, #

It is simply the candidate set before the intersection, with all insertion candidates removed.

MAXIO([-CONS]): The next candidate indicates that underlyingly [-CONS] segments cannot be deleted (here, 'V' represents the set of [-CONS] segments, *i.e.,* {a}), and is represented by the following WFSM:



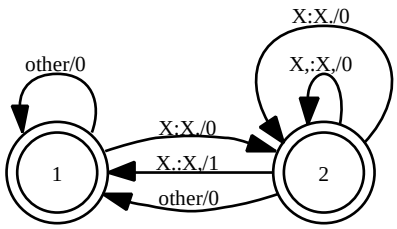
The candidate set that results from the intersection is as follows:

SR: # X X* ? ?* X X* #

UR: # a. a, h. h, a. a, #

In other words, the deletion option "-" has been removed from the surface correspondents for underlying /a/.

IO-INTEGRITY(X): This constraint ensures that the surface correspondent for each segment must be only one segment.



The resulting candidate set is as follows:

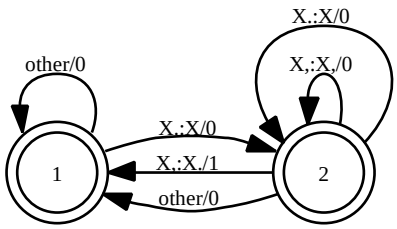
SR: # *X*. *X*,* - *X*. *X*,* #

UR: # a. a, h. a. a, #,

SR: # *X*. *X*,* *X* *X*,* *X* *X*,* #

UR: # a. a. h. h, a. a, #

IO-UNIFORMITY(X): The underlying correspondent for each segment must be only one segment.



The resulting candidate set is as follows:

SR: # *X*. *X*,* - *X*. *X*,* #

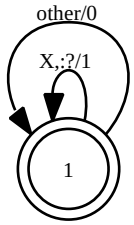
UR: # a. a, h. a. a, #,

SR: # *X*. *X*,* *X*. *X*,* *X*. *X*,* #

UR: # a. a, h. h, a. a, #

The only difference here from the previous candidate set is that the second underlying segment must correspond to a different surface segment from the first, and similarly with the third underlying segment.

*Contour: Bans contour segments (essentially, bans the continuation version of a segment, on the surface).



The result is as follows:

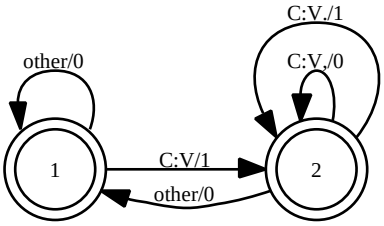
SR: # *X*. - *X*. #

UR: # a. h. a. #,

SR: # *X*. *X*. *X*. #

UR: # a. h. a. #

IdentIO([–cons]): Bans changing a vowel into a consonant.



The result is as follows:

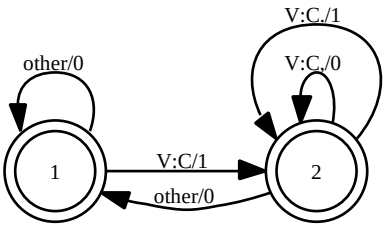
SR: # a. - a. #

UR: # a. h. a. #,

SR: # a. *X*. a. #

UR: # a. h. a. #

IDENTIO([+CONS]): Prevents changing a consonant into a vowel.



Result:

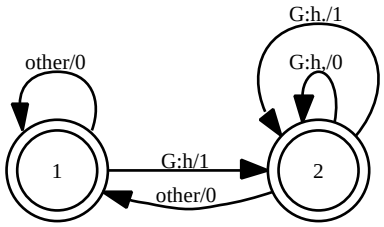
SR: # a. - a. #

UR: # a. h. a. #,

SR: # a. *C*. a. #

UR: # a. h. a. #

IDENTIO([+SPREAD-GLOTTIS]): Preserves [+spread-glottis]. Here 'h' represents the set of spread-glottis segments {h}, and 'G' represents the set of non-spread-glottis segments {a,b}.



Result:

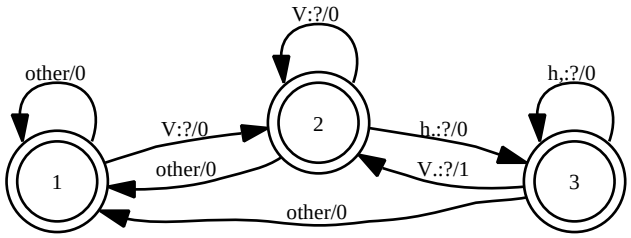
SR: # a. - a. #

UR: # a. h. a. #,

SR: # a. h. a. #

UR: # a. h. a. #

*DOUBLESEQUENCE([-CONS],[+SGLOT],[-CONS]): Bans [h] intervocalically (*VhV).



The result is as follows:

SR: # a. - a. #

UR: # a. h. a. #

The remaining constraints are, in order, DEPBR(C) (see below), which asserts that a consonant may not appear in the reduplicant without corresponding to a segment in the base, and MAXIO(C), which penalizes underlying consonants with no surface correspondent. The application of these other constraints will have no effect, as there is now only a single candidate.

2.5 REDUPLICATION

The next step, given the above example, is to figure out how to model McCarthy and Prince's analysis of over-applied forms such as [babaa] above. Their analysis is based upon the introduction of a new kind of correspondence – "Base-Reduplicant." In this analysis faithfulness relations are computed based on a correspondence between the surface form of the base and the surface form of the reduplicant. The problem with such a correspondence is that heretofore correspondence has been represented by vertical alignment. The options here are to introduce some sort of horizontal correspondence, *e.g.,* "a segment shall be faithful to the nth segment to its right," or to change the representation to somehow make the seemingly horizontal correspondence vertical. The first of these options has some promise, but also seems likely to be fairly complicated to implement in a general way, given word-length variation (the basic idea would be to have a separate constraint for each position in the base or something like that). It will be saved for possible future work.

Here I will describe an implementation of the second option.

The basic idea (Albro 1998a; Eisner 1997a) is that each candidate includes an exact copy of the reduplicant aligned vertically with the base. The inclusion of this reference copy allows constraints essentially identical to those used in the previous section to model Base-Reduplicant Correspondence. This benefit, however, comes at a price. The requirement that the candidate set consist only of candidates with two copies of the reduplicant puts this beyond the expressive power of finite state machines. The least complicated grammar type in the Chomsky Hierarchy that is able to represent such a set is a mildly context sensitive grammar. Unfortunately, the class of mildly context sensitive grammars is not closed with respect to intersection, so it is not possible for both the candidates and the constraints to be such grammars. It is a good thing that this is not one of our requirements, then. Luckily, mildly context sensitive grammars (or at least, the type I will be using — multiple context free grammars[4]) are closed with respect to intersection with finite state machines. The challenge, then, is to develop an algorithm for intersecting MCFGs with weighted finite state machines. This I have done, see Appendix C. The intersection algorithm has time complexity $O(n^6)$, which although still polynomial, is significantly slower than FSM intersection, which is $O(n^2)$ (both measured approximately in number of edges of the machine being intersected with the grammar).

[4] Examples of multiple context free grammars (MCFGs) will be given shortly; a formal definition is given in Appendix B.

2.5.1 MCFG INTERSECTION EXAMPLE

2.5.1.1 *An Example* CFG

The multiple context free grammar formalism is a fairly simple extension of the more well-known context free grammar formalism. Here is an example of a context-free grammar:

$$
\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{R} && (2.1)\\
\mathrm{R} &\rightarrow \mathrm{X}\\
\mathrm{R} &\rightarrow \mathrm{X\,R}\\
\mathrm{X} &\rightarrow \mathrm{A\,A}\\
\mathrm{X} &\rightarrow \mathrm{B\,B}\\
\mathrm{A} &\rightarrow \mathrm{a}\\
\mathrm{B} &\rightarrow \mathrm{b}
\end{aligned}
$$

Each of the lines in this grammar is called a *production*, or a *rewrite rule*. In a CFG the capitalized symbols are called *nonterminals* (only nonterminals appear on the left-hand side of a production) and the lowercase symbols are called *terminals*. A CFG represents the set of strings that can be *derived* from the start symbol S. A derivation proceeds by replacing nonterminal symbols by the right-hand side of some production they head, until no nonterminal symbols are left. The grammar above derives the language $(\mathtt{aa}|\mathtt{bb})^{+}$, that is, the set of strings consisting of one or more aa or bb sequences. It would include,

then, aa, bb, aaaa, aabb, bbaa, bbbb, etc. An example from this language is the string bbaabbbb, which is derived as follows:

1.	S	[*start symbol*]
2.	R	[S → R]
3.	XR	[R → XR]
4.	BBR	[X → BB]
5.	bBR	[B → b]
6.	bbR	[B → b]
7.	bbXR	[R → XR]
8.	bbAAR	[X → AA]
9.	bbaAR	[A → a]
10.	bbaaR	[A → a]
11.	bbaaXR	[R → XR]
12.	bbaaBBR	[X → BB]
13.	bbaabBR	[B → b]
14.	bbaabbR	[B → b]
15.	bbaabbX	[R → X]
16.	bbaabbBB	[X → BB]
17.	bbaabbbB	[B → b]
18.	bbaabbbb	[B → b]

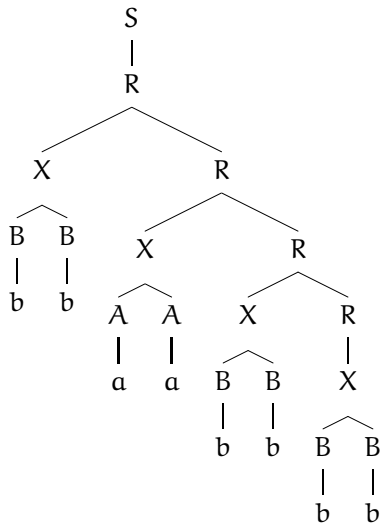


Figure 2.2: CFG derivation tree for bbaabbbb.

A derivation such as the above may be represented by a *derivation tree,* such as that shown in figure 2.2. Notice that the derived string can be read off by traversing the leaves of the tree from left to right. One more note about this derivation: the tree shows not only that bbaabbbb derives from S, but also that b derives from B, bb derives from X, aabbbb derives from R, *etc.*

A production such as

$$A \rightarrow B\ C$$

Can be read as an implication of the form

> If string β is derived from B and string γ is derived from C, then the concatenation $\beta\gamma$ of those strings is derived from A.

2.5.1.2 *An Example* MCFG

Now for a similar MCFG. This MCFG represents a simple language of totally reduplicated strings from the alphabet $\{a, b\}$. A compact representation of the language it represents might be $\{ww | w \in \{a, b\}^+\}$.

$$
\begin{aligned}
S &\rightarrow g_1[R], g_1(x_1) = (x_{11}x_{12}) \qquad (2.2)\\
R &\rightarrow g_2[X], g_2(x_1) = (x_{11}, x_{12})\\
R &\rightarrow g_3[X, R], g_3(x_1, x_2) = (x_{11}x_{21}, x_{12}x_{22})\\
X &\rightarrow g_4[A, A], g_4(x_1, x_2) = (x_{11}, x_{21})\\
X &\rightarrow g_4[B, B]\\
A &\rightarrow a\\
B &\rightarrow b
\end{aligned}
$$

Notice that this grammar looks just like the CFG given in (2.1) above, except that each *nonterminal* production (production with nonterminals in the right hand side) has a function associated with it. The function indicates what string (or group — "tuple" — of strings) is derived from a nonterminal given the (tuples of) strings derived from the terminals in the right-hand side. In a CFG, this function is always left-to-right concatenation, so if in a derivation the production $A \rightarrow B\,C$ is used where B derived abc and C derived def, then A derives $abcdef$. In an MCFG, however, the function may involve any concatenation of the derived strings, possibly grouping the strings into tuples

or ungrouping them. The functions are written in terms of subscripted variables such as x_{12}. These variables represent strings; the first subscript indicates which member of the rule's right-hand side the string comes from, and the second subscript indicates a particular element of the tuple of strings associated the selected right-hand side member. For example, in $R \rightarrow g_3[X, R], g_3(x_1, x_2) = (x_{11}x_{21}, x_{12}x_{22})$, X and R both derive pairs (2-tuples) of strings such as (a, a) or (b, b). In $g_3(x_1, x_2) = (x_{11}x_{21}, x_{12}x_{22})$, x_1 represents a pair of strings derived from the X in the right hand side of the production, and x_2 represents a pair of strings derived from R in the right hand side. The right-hand-side symbol x_{11} refers to the first element of a pair of strings derived from X; x_{12} to the second; x_{21} to the first element of a pair derived from R; and x_{22} to the second. Thus if it is previously known that (a, a) derives from X and (bb, bb) derives from R, $g_3(x_1, x_2) = g_3((a, a), (bb, bb)) = (x_{11}x_{21}, x_{12}x_{22}) = (abb, abb)$ and therefore (abb, abb) is derived from R.

These functions must mention all of the components of the right-hand side of the productions to which they are attached, and each component must be mentioned only once. Because of these functions, the MCFG represents a different language from the CFG ($\{ww|w \in \{a, b\}^+\}$ *vs.* $\{(aa|bb)^+\}$) despite having otherwise identical productions.

Figure 2.3 shows a derivation tree from the MCFG above; it is structurally identical to the tree in figure 2.2, but due to the string re-write functions of the MCFG it represents the derivation of an entirely different string. The obvious difference between this tree and the tree of figure 2.2 is that each node is labeled with a tuple of strings. This is not necessary in a CFG because the string derived at each node of a CFG derivation tree is computable from the tree without reference to the original grammar (the string derived from a given node

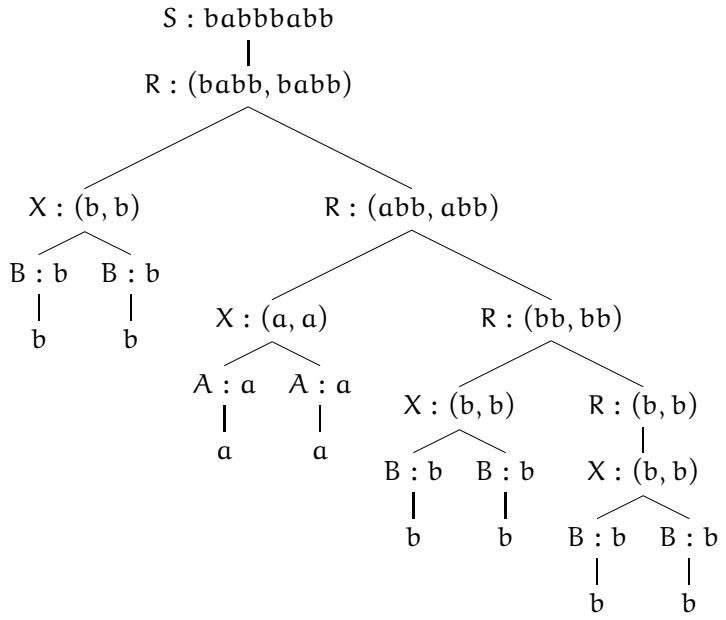


Figure 2.3: MCFG derivation tree for babbbabb

is the left-to-right concatenation of the labels of the leaves dominated by that node). The tree in figure 2.3 derives babbbabb despite its leaves reading, from left to right, bbaabbbb, which is, not so coincidentally, what the tree of figure 2.2 derives.

2.5.1.3 MCFG *for Base-Reduplicant Correspondence Theory Candidate Sets*

At this point the difference between an MCFG and a CFG should be reasonably clear. The process of computing Optimality Theoretic derivations using MCFGs to simulate Base-Reduplicant Correspondence Theory is not yet clear, however. The basic idea here is to have a candidate set in which each candidate has a copy of the reduplicant vertically aligned with the base such that correspondence relations can be computed. For example, the underlying form /RED+bah+e/ might have candidates like the following:

```
S: # b. a. -  b. a. h. e. #
U: # b. a. h. b. a. h. e. #
R: # -  -  -  b. a. -  -  #
M: # R. R, R, B. B, B, -  #
```

Here *S* represents the surface form, *U* the underlying form, *R* the reduplicant reference copy that is used for computing correspondence relations, and *M* represents a layer of morphological information. Inside *M*, "R. R, R," represents a reduplicant morpheme, "B. B, B," represents a base, and "–" represents no particular distinguished morpheme.

The candidate set properties described above are enforced by requiring that the candidate set be a subset of the language described by an MCFG that embodies the properties. This MCFG is a bit complex, so I will begin with a simplification of it—this simplification covers prefixing reduplication only, with no morphemes described other than the reduplicant and the base (*i.e.,* no prefixes or suffixes are allowed, other than the reduplicant itself). I will also, for now, leave out the word boundary symbols. Given these caveats,

here is an MCFG that describes the set of all candidates with four tiers as described above, where the reduplicant reference tier aligned with the base must contain an exact copy of whatever the surface tier aligned with the reduplicant contains:

$$
\begin{aligned}
S &\rightarrow g_1[R],\ g_1(x_1) = (x_{11}x_{12}) \\
R &\rightarrow g_2[R_i],\ g_2(x_1) = (x_{11}, x_{12}) \\
&\mid g_3[R_i, R_r],\ g_3(x_1, x_2) = (x_{11}x_{21}, x_{12}x_{22}) \\
R_i &\rightarrow g_3[R_a, M_{RBi}] \\
R_r &\rightarrow g_2[R_c] \\
&\mid g_3[R_c, R_r] \\
R_c &\rightarrow g_3[R_a, M_{RBc}] \\
R_a &\rightarrow g_4[X, O]\ g_4(x_1, x_2) = (x_{11}x_{21}, x_{22}x_{12}) \\
X &\rightarrow g_2[C_{ai}, C_{ai}] \\
&\mid g_2[C_{ac}, C_{ac}] \\
&\mid g_2[C_{bi}, C_{bi}] \\
&\vdots \\
C_{ai} &\rightarrow \texttt{a}. \\
C_{ac} &\mid \texttt{a}, \\
&\vdots \\
O &\rightarrow g_5[O_{ur}, O_{su}],\ g_5(x_1, x_2) = (x_{11}, x_{21}) \\
O_{ur} &\rightarrow g_6[A, C_{\_}],\ g_6(x_1, x_2) = (x_{11}x_{21})
\end{aligned}
\tag{2.3}
$$

$$
\begin{aligned}
O_{su} &\rightarrow g_6[A, A] \\
A &\rightarrow \texttt{a.} \\
&\mid \texttt{a,} \\
&\mid \texttt{b.} \\
&\vdots \\
C_{\_} &\rightarrow \_ \\
M_{RBi} &\rightarrow g_5[C_{Ri}, C_{Bi}] \\
M_{RBc} &\rightarrow g_5[C_{Rc}, C_{Bc}] \\
C_{Ri} &\rightarrow \texttt{R.} \\
C_{Bi} &\rightarrow \texttt{B.} \\
C_{Rc} &\rightarrow \texttt{R,} \\
C_{Bc} &\rightarrow \texttt{B,}
\end{aligned}
$$

In words, the grammar says that a candidate (S) consists of a reduplicated form (R), where a reduplicated form is a pair of strings. These strings are concatenated (reduplicant, then base) to make a candidate. A reduplicated form

$$
\begin{aligned}
R &\rightarrow g_2[R_i],\ g_2(x_1) = (x_{11}, x_{12}) \\
&\mid g_3[R_i, R_r],\ g_3(x_1, x_2) = (x_{11}x_{21}, x_{12}x_{22}) \\
R_i &\rightarrow g_3[R_a, M_{RBi}] \\
R_r &\rightarrow g_2[R_c] \\
&\mid g_3[R_c, R_r] \\
R_c &\rightarrow g_3[R_a, M_{RBc}]
\end{aligned}
$$

$$R_a \quad \rightarrow \quad g_4[X, O]\ g_4(x_1, x_2) = (x_{11}x_{21}, x_{22}x_{12})$$

consists of an initial reduplicated form (R_i), optionally followed by the continuation reduplicated form (R_r). R_i represents the underlined parts of the example form in the following:

S: b. a. - b. a. h.

U: b. a. h. b. a. h.

R: - - - b. a. -

M: R. R, R, B. B, B,

R_r then represents the remainder, each vertical slice of which is represented by R_c. Notice that R_i and R_c are not recursive, but R_r is. Each of R_i and R_c contains R_a. R_a is the component that ensures that the reduplicant reference tier contains within the base an exact copy of the surface form of the reduplicant. It consists of a pair X of identical elements, plus O, an amalgam of the stuff that does not require dependency. It is then arranged such that the first element of each X is the surface element of one vertical slice in the reduplicant and the second (identical) element is the reduplicant reference-tier element of a corresponding slice of the base. A partial derivation tree for the /RED+bah/ example above is given in Figure 2.4. For reasons of space I was unable to include the derived text at each node as in Figure 2.3. Instead, I included the indices of the derived text, in square brackets (so, for example, "[2,4]" would indicate "- R.", the reduplicant reference and morphological elements of the first vertical slice, and [0,24] would indicate the entire example).

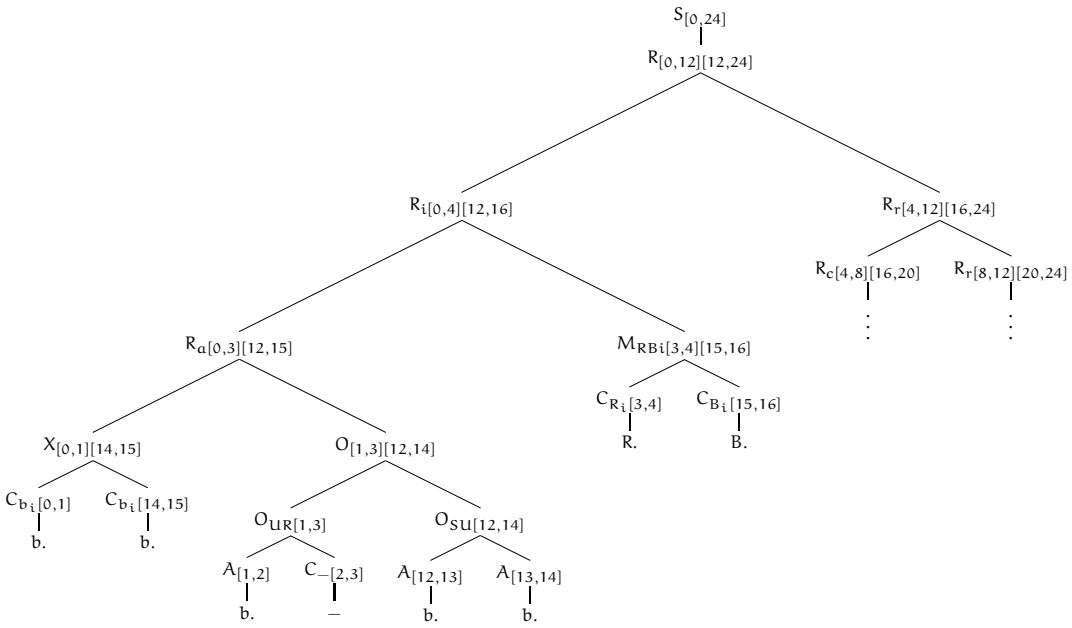


Figure 2.4: Derivation tree for /RED+bah/

The full grammar adds word boundaries, prefixes, and suffixes, and supports both prefixing and suffixing reduplication. It is as follows:

$$
\begin{aligned}
S &\rightarrow g_1[B_{w4}, W_{ir}],\ g_1(x_1, x_2) = (x_{11}x_{21}) && (2.4)\\
B_{w4} &\rightarrow g_1[B_{w2}, B_{w2}]\\
W_{ir} &\rightarrow g_1[W_i, B_{w4}]\\
B_{w2} &\rightarrow g_1[B_w, B_w]\\
B_w &\rightarrow \#\\
W_i &\rightarrow g_1[F, W_{i2}]\\
&\quad | \quad g_2[W_{i2}],\ g_2(x_1) = (x_{11})\\
&\quad | \quad g_2[F]
\end{aligned}
$$

$$
\begin{aligned}
W_{i2} &\rightarrow g_1[W_r, F] \\
&\mid g_2[W_r] \\
W_r &\rightarrow g_3[R_p],\ g_3(x_1) = (x_{11}x_{12}) \\
&\mid g_3[R_s] \\
R_p &\rightarrow g_4[R_{pi}],\ g_4(x_1) = (x_{11}, x_{12}) \\
&\mid g_5[R_{pi}, R_{pr}],\ g_5(x_1, x_2) = (x_{11}x_{21}, x_{12}x_{22}) \\
R_s &\rightarrow g_4[R_{si}] \\
&\mid g_5[R_{si}, R_{sr}] \\
R_{pi} &\rightarrow g_5[R_{pa}, M_{RBi}] \\
R_{si} &\rightarrow g_5[R_{sa}, M_{BRi}] \\
R_{pr} &\rightarrow g_4[R_{pc}] \\
&\mid g_5[R_{pc}, R_{pr}] \\
R_{sr} &\rightarrow g_4[R_{sc}] \\
&\mid g_5[R_{sc}, R_{sr}] \\
R_{pc} &\rightarrow g_5[R_{pa}, M_{RBc}] \\
R_{sc} &\rightarrow g_5[R_{sa}, M_{BRc}] \\
R_{pa} &\rightarrow g_6[X, O],\ g_6(x_1, x_2) = (x_{11}x_{21}, x_{22}x_{12}) \\
R_{sa} &\rightarrow g_7[O, X],\ g_7(x_1, x_2) = (x_{12}x_{21}, x_{22}x_{11}) \\
X &\rightarrow g_8[C_{ai}, C_{ai}],\ g_8(x_1, x_2) = (x_{11}, x_{21}) \\
&\mid g_8[C_{ac}, C_{ac}] \\
&\mid g_8[C_{bi}, C_{bi}]
\end{aligned}
$$

$$
\begin{aligned}
&\vdots \\
C_{ai} &\rightarrow \texttt{a}. \\
C_{ac} &\mid \texttt{a}, \\
&\vdots \\
O &\rightarrow g_8[O_{ur}, O_{su}] \\
O_{ur} &\rightarrow g_1[A, C_{\_}] \\
O_{su} &\rightarrow g_1[A, A] \\
A &\rightarrow \texttt{a}. \\
&\mid \texttt{a}, \\
&\mid \texttt{b}. \\
&\vdots \\
C_{\_} &\rightarrow \_ \\
M_{RBi} &\rightarrow g_8[C_{Ri}, C_{Bi}] \\
M_{RBc} &\rightarrow g_8[C_{Rc}, C_{Bc}] \\
M_{BRi} &\rightarrow g_8[C_{Bi}, C_{Ri}] \\
M_{BRc} &\rightarrow g_8[C_{Bc}, C_{Rc}] \\
C_{Ri} &\rightarrow \texttt{R}. \\
C_{Bi} &\rightarrow \texttt{B}. \\
C_{Rc} &\rightarrow \texttt{R}, \\
C_{Bc} &\rightarrow \texttt{B},
\end{aligned}
$$

$$
\begin{aligned}
F &\rightarrow g_2[F_a] \\
&\mid\ g_1[F_a, F] \\
F_a &\rightarrow g_1[O_{su}, O_{mm}] \\
O_{mm} &\rightarrow g_1[C_{\_}, C_{\_}]
\end{aligned}
$$

2.5.1.4 *Ground Covered So Far*

To summarize the ground covered so far, I have shown how weighted finite state Optimality Theory can work, including a system for constraint and candidate representation, and I have introduced the multiple context-free grammar formalism and shown how it can delimit a set of candidates that have a reference copy of the reduplicant aligned with the base such that the same sort of constraint that enforces input-output correspondence relations can enforce base-reduplicant correspondence relations. What remains is to show how to take the candidate set produced by Gen, which is expressed as a finite state machine, and modify it such that it obeys the restrictions of the grammar given in (2.4) above, and, further, how to apply the winnowing process to the resulting candidate set.

The output of Gen is a grammar that describes a set of candidates. The grammar of (2.4), above, also describes a set of candidates. What is desired then is the set of candidates described by both grammars, *i.e.,* the intersection of the Gen candidate set with the grammar's candidate set. Luckily, it is possible to find such an intersection by a modification of *deductive chart parsing* (Shieber *et al.* 1995).

2.5.1.5 *Bottom-Up* CFG *Chart Parsing*

The goal is to show how bottom-up chart parsing of multiple context free grammars can be used to intersect a candidate set—expressed as a finite state machine—with the reference-identity requirement for the Base-Reduplicant Correspondence Theoretic representation—expressed as a multiple context free grammar. To do this, it is first necessary to investigate what bottom-up chart parsing is. The full algorithm for bottom-up deductive chart parsing of multiple context free grammars is somewhat complicated, so I will first discuss a simpler, and more standard, application of bottom-up deductive chart parsing. Deductive chart parsing is most simply applied to prove whether or not a given string is a member of the language described by a grammar. I will begin with context free grammars since they are simpler than multiple context free grammars, and show how this process is applied to a string taken from the context free grammar given in (2.1), repeated here as (2.5):

$$
\begin{aligned}
S &\rightarrow R \\
R &\rightarrow X \\
R &\rightarrow X\,R \\
X &\rightarrow A\,A \\
X &\rightarrow B\,B \\
A &\rightarrow a \\
B &\rightarrow b
\end{aligned}
\tag{2.5}
$$

The string is part of the same one that was derived earlier: bbaa.

Deductive chart parsing is a variant of chart parsing that views the parsing process as a logical proof in a simple system. The chart, here, is a database of known facts, called *items*. Each item in a bottom-up context free grammar parse consists of a single grammatical symbol (*i.e.,* a terminal or a non-terminal) plus an indication of the extent of the string to be parsed that is covered by the symbol. In the example string, the positions in the string are numbered as follows: ${}_0b_1b_2a_3a_4$. Thus 0 is the initial position in the string and 4 is the final position. Using this numbering, an item looks like $\gamma[p, q]$, where p and q are indices chosen such that $0 \leq p < q \leq 4$, and γ represents some terminal or nonterminal symbol from the grammar employed. $\gamma[p, q]$ indicates that the subsequence of the string bounded by p, q can be derived from γ in zero or more steps (a terminal symbol derives itself in zero steps, so one item might be $b[0, 1]$). For example, $X[0, 2]$ indicates that X derives the initial bb of the string.

I mentioned deduction and proof. Deductive chart parsing, like all systems of logical deduction, begins with a set of *axioms* — things known at the beginning of the process — and a set of deductive rules used to establish new derived facts. The axioms in chart parsing are the words of the sentence being parsed, in this case $b[0, 1]$, $b[1, 2]$, $a[2, 3]$, $a[3, 4]$. The deductive rules are related to the rules of grammar:

$$\frac{A[i, j]B[j, k]}{C[i, k]} \; \textit{if} \; C \rightarrow AB \in G \tag{2.6}$$

$$\frac{A[i, j]}{C[i, j]} \; \textit{if} \; C \rightarrow A \in G \tag{2.7}$$

The interpretation of a deductive rule is that if a set of facts from the set of known facts matches the top part of the rule (the *antecedents),* then the rule applies and the facts in the bottom part of the rule (the *consequents)* are deduced. In deductive chart parsing there

are two places where facts are stored: a *stack* and a *chart*. The stack is like the in-box on a desk — it contains a pile of items to be worked on; only the top item in the stack can be removed from it at a given time. The chart contains all known facts, all of which are accessible at all times.

To summarize the above, bottom-up deductive chart parsing for context free grammars works as follows:

1. Add all axiomatic items (one for each terminal symbol in the sentence being parsed) to the stack and the chart.

2. Take the top item from the stack. If this item matches a deductive rule (by itself or with any other item in the chart), *i.e.* if it is in the right-hand side of some grammar production, add the item deduced (the left-hand side of the production, with string-coverage marked) to the top of the stack and to the chart.

3. Repeat step 2 until the stack is empty.

If at the end of the process above a *goal* item has been added to the chart, the sentence being parsed has been proved to be a member of the language of the grammar being used. A goal item is of the form $S[0, f]$, where S is the start symbol of the grammar and f is the length of the sentence being parsed.

In the example, the initial chart contains $b[0, 1]$, $b[1, 2]$, $a[2, 3]$, $a[3, 4]$ and the stack is the same. Removing the top item, $b[0, 1]$, the deductive rule

$$\frac{b[0, 1]}{B[0, 1]}$$

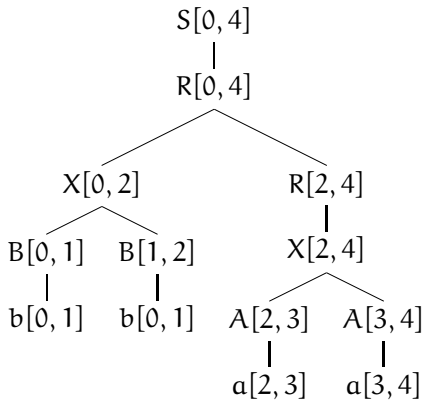
applies, since $B \rightarrow b$ is a production, so $B[0,1]$ is added to the chart and the stack becomes $B[0,1], b[1,2], a[2,3], a[3,4]$. B is in the right-hand side of $X \rightarrow BB$, but

$$\frac{B[0,1]B[0,1]}{X[0,1]}$$

does not match

$$\frac{B[i,j]B[j,k]}{X[i,k]}$$

since $0 \neq 1$. Therefore no rule applies to $B[0,1]$ so we proceed to $b[1,2]$ and get $B[1,2]$, the stack becoming $B[1,2], a[2,3], a[3,4]$. It is now possible to apply the $X \rightarrow BB$ rule properly. $B[0,1]$ from the chart and $B[1,2]$ from the stack combine to form $X[0,2]$, and the stack becomes $X[0,2], a[2,3], a[3,4]$. From $X[0,2]$ we derive $R[0,2]$—$R[0,2], a[2,3], a[3,4]$, and from $R[0,2]$ we derive $S[0,2]$, which is not a goal item and which derives nothing else, so we move on to $a[2,3]$. From $a[2,3]$ we get $A[2,3]$—$A[2,3], a[3,4]$. $A[2,3]$ by itself cannot make an X, so we move on to $a[3,4]$, which yields $A[3,4]$. From $A[2,3]$ and $A[3,4]$ we get $X[2,4]$. The stack now contains only $X[2,4]$, from which we get $R[2,4]$. From $R[2,4]$ we derive two items: $S[2,4]$ via $S \rightarrow R$, and $R[0,4]$, via $R \rightarrow XR$ with $X[0,2]$. $S[2,4]$ is a dead end, but $R[0,4]$ derives $S[0,4]$, which is the goal item. Therefore $bbaa$ is an element of the language. The derivation tree,

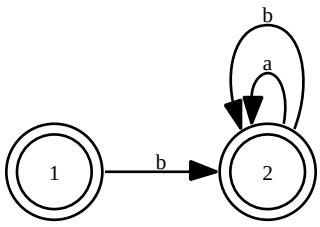


may be reconstructed from the chart by working backwards from S[0, 4]. The process works something like this:

> In the grammar S heads only one production: S → R, so from S[0, 4] we must derive R[0, 4]. That is in the chart, so add it to the tree. R then heads both R → X and R → XR, but there is no X[0, 4] in the chart. The only Rs in the chart that end in 4 are R[0, 4] and R[2, 4]. There is no X in the chart ending with 0, but there is an X ending with 2, X[0, 2]. Therefore R[0, 4] derives X[0, 2], R[2, 4] in the tree...

The process continues from there, building the tree top-down by looking up elements in the chart to find anything that could complete a production.

Chart Parsing with Finite State Machines Earlier it was promised that chart parsing could be used to intersect a finite state machine with a multiple context-free grammar. In approaching that goal it may be helpful to consider a finite state machine-context free grammar intersection example. Take the following FSM:



To intersect this machine with the context-free grammar in (2.5), it is possible to use almost exactly the same algorithm as was used to parse the sentence bbaa. The main difference is that the axioms are drawn from the arcs of the FSM rather than the words

of the sentence. The indices used to label the items in the chart are no longer word numbers within the sentence, but rather state numbers from the finite state machine. Thus, the algorithm begins with the axioms $b[1,2]$, $a[2,2]$, and $b[2,2]$ in both the stack and the chart. The goal item here is $S[1,f]$, where f is any final state (here, either 1 or 2). The parse then proceeds much as the earlier parse did. The top item of the stack, $b[1,2]$, yields $B[1,2]$— $B[1,2]$,$a[2,2]$, $b[2,2]$. $B[1,2]$ yields nothing by itself, so the parse proceeds to $a[2,2]$, which yields $A[2,2]$—$A[2,2]$, $b[2,2]$. From $A[2,2]$ we get $X[2,2]$ (via $X[2,2] \rightarrow A[2,2]A[2,2]$), and thence $R[2,2]$, which leads to $S[2,2]$. Further, $R[2,2]$ plus $X[2,2]$ yields $R[2,2]$, but that is already in the chart. The stack now contains $b[2,2]$. From there we get $B[2,2]$, which yields both $X[1,2]$ via $X[1,2] \rightarrow B[1,2]B[2,2]$ and $X[2,2]$ via $X[2,2] \rightarrow B[2,2]B[2,2]$. From $X[1,2]$ we get $R[1,2]$ both directly ($R[1,2] \rightarrow X[1,2]$) and via $R[1,2] \rightarrow X[1,2]R[2,2]$. Finally, $R[1,2]$ gives us $S[1,2]$, which is the goal item. Reconstructing the intersection grammar, we get

$$
\begin{aligned}
S[1,2] &\rightarrow R[1,2] \\
R[1,2] &\rightarrow X[1,2] \\
&\quad\ |\ X[1,2]\ R[2,2] \\
R[2,2] &\rightarrow X[2,2] \\
&\quad\ |\ X[2,2]\ R[2,2] \\
X[1,2] &\rightarrow B[1,2]\ B[2,2] \\
X[2,2] &\rightarrow B[2,2]\ B[2,2] \\
&\quad\ |\ A[2,2]\ A[2,2] \\
B[1,2] &\rightarrow b[1,2]
\end{aligned}
$$

$$A[2,2] \rightarrow a[2,2]$$

$$B[2,2] \rightarrow b[2,2]$$

This can be simplified slightly to the following:

$$S \rightarrow R_1$$

$$R_1 \rightarrow X_1 | X_1\ R_2$$

$$R_2 \rightarrow X_2 | X_2\ R_2$$

$$X_1 \rightarrow B\ B$$

$$X_2 \rightarrow A\ A | B\ B$$

$$A \rightarrow a$$

$$B \rightarrow b$$

The language of this grammar is $bb(aa|bb)^*$, which is indeed the intersection of $b(a|b)^*$ (the finite state language) with $(bb|aa)^+$ (the context-free language).

2.5.1.6 MCFG *Chart Parsing*

Up to this point it has been shown how to intersect a finite state machine with a CFG and recover the intersection grammar (also a CFG). From here all that remains is to show how to extend this to intersecting finite state machines with multiple context free grammars, and finally to extend *that* to intersection of a multiple context free grammar with a *weighted* finite state machine. The first extension allows for adding the reduplicant reference property to a candidate set, and the second extension allows for winnowing said candidate set via finite state constraints.

To review, a CFG of the normal form used above (a *normal form* is a restriction in the way a grammar is written that does not affect the expressive power of the grammar) has three types of productions:

$$\begin{aligned} A &\rightarrow a \\ B &\rightarrow C \\ D &\rightarrow E\,F \end{aligned}$$

Expressed as an MCFG, the same thing would be

$$\begin{aligned} A &\rightarrow a \\ B &\rightarrow g_1[C],\ g_1(x_1) = (x_{11}) \\ D &\rightarrow g_2[E, F],\ g_2(x_1, x_2) = (x_{11}x_{21}) \end{aligned}$$

In words, both grammars can be interpreted as follows:

1. The string a is of category A. Alternatively, A derives a, a parses as A, a is in the yield of A.

2. Any string that is of category C is also of category B.

3. If string x_1 is of category E and string x_2 is of category F, then their concatenation x_1x_2 is of category D.

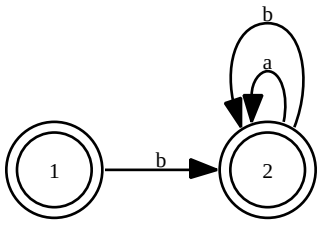
An MCFG adds some new possibilities, as the string-combination function on the right hand side of a production may be something other than simple left-to-right concatenation. It can also do two other things: first, it may reorder the substrings, and, second, it

may group strings into tuples or extract strings from tuples, as has been seen. The question, then, is how to deal with this added complexity. Looking at the items from the chart parsing example given earlier, each consists of a grammatical symbol plus a pair of numbers that indicates a subsequence of the input sentence, or a sub-FSM consisting of paths between two states. For MCFGs, we extend the definition of an item to include a tuple of such pairs instead of just one pair, and the pairs combine to form new items according to the string function of the production being used. The usual rule applies that $[p, q]$ concatenates with $[r, s]$ only if q and r are the same. The exact details of the extension are given in Appendix C (see also Albro (2002) for an alternative method), but an example should help to make the idea clear.

Let us take the simple reduplication grammar given before:

$$
\begin{aligned}
S &\rightarrow g_1[R],\ g_1(x_1) = (x_{11}x_{12}) \qquad (2.8)\\
R &\rightarrow g_2[X],\ g_2(x_1) = (x_{11}, x_{12})\\
R &\rightarrow g_3[X, R],\ g_3(x_1, x_2) = (x_{11}x_{21}, x_{12}x_{22})\\
X &\rightarrow g_4[A, A],\ g_4(x_1, x_2) = (x_{11}, x_{21})\\
X &\rightarrow g_4[B, B]\\
A &\rightarrow a\\
B &\rightarrow b
\end{aligned}
$$

As before, we intersect with



I will show the intersection via the stack, plus the production used:

1.	$b[1,2], a[2,2], b[2,2]$	*axioms*
2.	$B[1,2], a[2,2], b[2,2]$	$B \to b$
3.	$X[1,2][1,2], a[2,2], b[2,2]$	$X \to g_4[B,B], g_4(x_1,x_2) = (x_{11}, x_{21})$
4.	$R[1,2][1,2], a[2,2], b[2,2]$	$R \to g_2[X], g_2(x_1) = (x_{11}, x_{12})$
5.	$A[2,2], b[2,2]$	$A \to a$
6.	$X[2,2][2,2], b[2,2]$	$X \to g_4[A,A]$
7.	$R[2,2][2,2], b[2,2]$	$R \to g_2[X]$
8.	$B[2,2]$	$B \to b$
9.	$X[1,2][2,2], X[2,2][1,2]$	$X \to g_4[B,B]$
10.	$R[1,2][2,2], X[2,2][1,2]$	$R \to g_2[X], R \to g_3[X,R]$, $g_3(x_1,x_2) = (x_{11}x_{21}, x_{12}x_{22})$
11.	$S[1,2], X[2,2][1,2]$	$S \to g_1[R], g_1(x_1) = (x_{11}x_{12})$
12.	$R[2,2][1,2]$	$R \to g_2[X], R \to g_3[X,R]$

Some notes — after step 4 we do not produce an element of S from $R[1,2][1,2]$ because this would involve concatenating $[1,2]$ with $[1,2]$, and 1 is not equal to 2. After step 7, $R[2,2][2,2]$ combines with $X[2,2][2,2]$ to produce $R[2,2][2,2]$, as well as with $X[1,2][1,2]$

to produce $R[1,2][1,2]$, but these are already in the chart, so the stack is not affected. After step 8, $B[2,2]$ could produce $X[2,2][2,2]$,but it is already in the chart. Step 9 uses $B[1,2]$ and $B[2,2]$. Step 10 produces $R[1,2][2,2]$ either via $X[1,2][2,2]$ directly or the combination of $X[1,2][2,2]$ with $R[2,2][2,2]$. From the chart it is possible to reconstruct the intersection grammar in the same manner as before:

$$
\begin{aligned}
S[1,2] &\rightarrow g_1[R[1,2][2,2]] \\
R[1,2][2,2] &\rightarrow g_2[X[1,2][2,2]] \\
&\quad | \quad g_3[X[1,2][2,2], R[2,2][2,2]] \\
R[2,2][2,2] &\rightarrow g_2[X[2,2][2,2]] \\
&\quad | \quad g_3[X[2,2][2,2]], R[2,2][2,2]] \\
X[1,2][2,2] &\rightarrow g_4[B[1,2], B[2,2]] \\
X[2,2][2,2] &\rightarrow g_4[A[2,2], A[2,2]] \\
&\quad | \quad g_4[B[2,2], B[2,2]] \\
A[2,2] &\rightarrow a \\
B[1,2] &\rightarrow b \\
B[2,2] &\rightarrow b
\end{aligned}
$$

Simplified, the grammar appears as follows:

$$
\begin{aligned}
S &\rightarrow g_1[R_1] \\
R_1 &\rightarrow g_2[X_1] \\
&\quad | \quad g_3[X_1, R_2] \\
R_2 &\rightarrow g_2[X_2]
\end{aligned}
$$

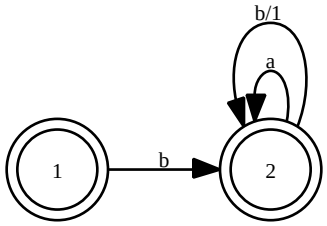
$$
\begin{aligned}
&| \quad g_3[X_2, R_2] \\
X_1 \quad &\rightarrow \quad g_4[B, B] \\
X_2 \quad &\rightarrow \quad g_4[A, A] \mid g_4[B, B] \\
A \quad &\rightarrow \quad a \\
B \quad &\rightarrow \quad b
\end{aligned}
$$

The language of this grammar is $\{ww|w \in b(a|b)^*\}$, which is indeed the intersection of $\{ww|w \in (a|b)^+\}$ with $b(a|b)^*$.

At this point all of the necessary steps are in place to convert a finite state candidate set into a multiple context free grammar with a proper reduplicant reference tier. Simply replace the simple grammar used here with the more complex one given in (2.4). After this has been accomplished, however, it is still necessary to intersect this candidate set with a series of constraints, which are here represented as weighted finite state machines. To achieve this a simple extension of the MCFG-FSM intersection is necessary. This involves adding to each item in the chart a *weight*. There are three ways of creating items in the chart, and each must be modified to deal with weights. First of all, the weight of an axiomatic item is just the weight of the arc from which the item is derived. Second, the weight of an item derived by a rule with a single element in its right hand side is just the weight of the item that corresponds to that single element. Finally, the weight of an item derived by a rule of the form $A \rightarrow g[B, C]$, that is, with two right-hand-side elements, is the sum of the weights of the two items that combined to form it. In the original algorithm, an item is added to the chart whenever no equivalent item is already in the chart. In this update, an item is added whenever no equivalent or lower-weight equivalent

item is already there. This modification is necessary to keep getting in an infinite loop uselessly adding higher and higher weight versions of the same item. When reconstructing the intersection grammar, one starts with the lowest-weighted goal elements and then proceeds as before, making sure that weights sum correctly.

As an example, take the finite state machine from the last example and add a weight:



The intersection of this machine with the simple reduplication grammar of (2.8) proceeds as follows:

1.	$b[1,2]:0, a[2,2]:0, b[2,2]:1$	*axioms*
2.	$B[1,2]:0, a[2,2]:0, b[2,2]:1$	$B \rightarrow b$
3.	$X[1,2][1,2]:0, a[2,2]:0, b[2,2]:1$	$X \rightarrow g_4[B,B], g_4(x_1,x_2) = (x_{11}, x_{21})$
4.	$R[1,2][1,2]:0, a[2,2]:0, b[2,2]:1$	$R \rightarrow g_2[X], g_2(x_1) = (x_{11}, x_{12})$
5.	$A[2,2]:0, b[2,2]:1$	$A \rightarrow a$
6.	$X[2,2][2,2]:0, b[2,2]:1$	$X \rightarrow g_4[A,A]$
7.	$R[2,2][2,2]:0, b[2,2]:1$	$R \rightarrow g_2[X]$
8.	$B[2,2]:1$	$B \rightarrow b$
9.	$X[1,2][2,2]:1, X[2,2][1,2]:1$	$X \rightarrow g_4[B,B]$
10.	$R[1,2][2,2]:1, X[2,2][1,2]:1$	$R \rightarrow g_2[X], R \rightarrow g_3[X,R],$ $g_3(x_1,x_2) = (x_{11}x_{21}, x_{12}x_{22})$

11.	$S[1,2]:1, X[2,2][1,2]:1$	$S \to g_1[R], g_1(x_1) = (x_{11}x_{12})$
12.	$R[2,2][1,2]:1$	$R \to g_2[X], R \to g_3[X,R]$

The derivation process is essentially the same as before, except that weights are added. Notice that we avoided combining $B[2,2] : 1$ with itself to form $X[2,2][2,2] : 2$. The reconstructed grammar appears as follows:

$$
\begin{aligned}
S[1,2]:1 &\to g_1[R[1,2][2,2]:1] \\
R[1,2][2,2]:1 &\to g_2[X[1,2][2,2]:1] \\
&\mid g_3[X[1,2][2,2]:1, R[2,2][2,2]:0] \\
R[2,2][2,2]:0 &\to g_2[X[2,2][2,2]:0] \\
&\mid g_3[X[2,2][2,2]:0], R[2,2][2,2]:0] \\
X[1,2][2,2]:1 &\to g_4[B[1,2]:0, B[2,2]:1] \\
X[2,2][2,2]:0 &\to g_4[A[2,2]:0, A[2,2]:0] \\
A[2,2]:0 &\to a \\
B[1,2]:0 &\to b \\
B[2,2]:1 &\to b
\end{aligned}
$$

It can then be simplified to remove weights and such:

$$
\begin{aligned}
S &\to g_1[R_1] \\
R_1 &\to g_2[X_1]
\end{aligned}
$$

$$
\begin{aligned}
&| && g_3[X_1, R_2] \\
R_2 &\rightarrow && g_2[X_2] \\
&| && g_3[X_2, R_2] \\
X_1 &\rightarrow && g_4[B, B] \\
X_2 &\rightarrow && g_4[A, A] \\
A &\rightarrow && a \\
B &\rightarrow && b
\end{aligned}
$$

So, interestingly, although the intersection process was almost identical, the resulting grammar is different — it represents the language $\{ww|w \in ba*\}$ rather than $\{ww|w \in b(a|b)*\}$ as before.

2.5.2 JAVANESE /H/-DELETION UNDER REDUPLICATION

With this machinery in place, we can resume the pseudo-Javanese example from §2.4, looking now at some of the reduplication examples.

2.5.2.1 *Simple Reduplication*

Consider first a fairly transparent example where the reduplicant looks like the base and the non-reduplicating phonology of the language applies: /RED+ bah/ = [bahbah]. The output of GEN is the following:

SR: # X ? ?* X* ? ?* X* ? ?* X* ? ?* X* ? ?* X* ? ?* X* #*

UR: # - b. b, - a. a, - h. h, - b. b, - a. a, - h. h, - #

RR: # *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, R, R, R, R, R, R, *?* B. B, B, B, B, B, B, B, B, #

(Note – I've left the morphological tier inaccurate for reasons of space. The actual idea is that all candidates have one 'R.' followed by zero or more 'R,', followed by one 'B.', followed by zero or more 'B,', and that 'R.' is the first element of the tier in all candidates and 'B.' starts either aligned with the initial underlying base 'b.' or somewhere in the insertion interval just before it). The candidate set here is represented by a finite state machine, as before. For reasons of efficiency, the candidate set is not converted into an MCFG until it is absolutely necessary, *i.e.,* until just before the first Base-Reduplicant correspondence constraint must be applied to the candidate set.

The constraints are the same as before, in the same ranking.

DEPIO(X): The highest-ranked constraint ensures that nothing is inserted, so the candidate set after winnowing is much simplified:

SR: # *?* *?** *?* *?** *?* *?** *?* *?** *?* *?** *?* *?** #

UR: # b. b, a. a, h. h, b. b, a. a, h. h, #

RR: # *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, R, R, R, B. B, B, B, B, B, #

MaxIO([-cons]): Next, the possibility of vowel deletion is dispensed with, and some ?'s become *X*'s.

SR: # *?* *?** *X* *X** *?* *?** *?* *?** *X* *X** *?* *?** #

UR: # b. b, a. a, h. h, b. b, a. a, h. h, #

RR: # *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, R, R, R, B. B, B, B, B, B, #

MaxIO([-spread-glottis]): This constraint is not necessary, but it makes no difference to the outcome and helps make presentation of the candidate sets bearable. After it, only /h/ can be deleted in candidates.

SR: # *X.* *X** *X* *X** *?* *?** *X* *X** *X* *X** *?* *?** #

UR: # b. b, a. a, h. h, b. b, a. a, h. h, #

RR: # *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, R, R, R, B. B, B, B, B, B, #

IO-INTEGRITY(X): This constraint ensures that no underlying segment corresponds to more than one surface segment. Essentially, it prevents surface members of the "initials" (dot) set of segments from corresponding to underlying members of the "continuation" (comma) set. A representation using surface "?" symbols is no longer possible, so I now give four distinguished subsets depending on whether or not underlying /h/ is preserved.

SR: # *X*. *X*,* *X* *X*,* - *X*. *X*,* *X* *X*,* - #

UR: # b. b, a. a, h. b. b, a. a, h. #

RR: # *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, R, R, B. B, B, B, B, #,

SR: # *X*. *X*,* *X* *X*,* - *X*. *X*,* *X* *X*,* *X* *X*,* #

UR: # b. b, a. a, h. b. b, a. a, h. h, #

RR: # *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, R, R, B. B, B, B, B, B, #,

SR: # *X*. *X*,* *X* *X*,* *X* *X*,* *X* *X*,* *X* *X*,* - #

UR: # b. b, a. a, h. h, b. b, a. a, h. #

RR: # *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, R, R, R, B. B, B, B, B, #,

SR: # *X*. *X*,* *X* *X*,* *X* *X*,* *X* *X*,* *X* *X*,* *X* *X*,* #

UR: # b. b, a. a, h. h, b. b, a. a, h. h, #

RR: # *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, R, R, R, B. B, B, B, B, B, #

IO-UNIFORMITY(X): This constraint discourages instances wherein two or more underlying segments correspond to a single surface segments, *i.e.,* instances of coalescence. It prevents surface members of the "continuation" (comma) set of segments from corresponding to underlying members of the "initial" (dot) set. At this point, each underlying segment now corresponds to either no surface segment, or exactly one.

SR: # *X. X,* X. X,** - *X. X,* X. X,** - #

UR: # b. b, a. a, h. b. b, a. a, h. #

RR: # *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, R, R, B. B, B, B, B, #,

SR: # *X. X,* X. X,** - *X. X,* X. X,* X. X,** #

UR: # b. b, a. a, h. b. b, a. a, h. h, #

RR: # *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, R, R, B. B, B, B, B, B, #,

SR: # *X. X,* X. X,* X. X,* X. X,* X. X,** - #

UR: # b. b, a. a, h. h, b. b, a. a, h. #

RR: # *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, R, R, R, B. B, B, B, B, #,

SR: # *X. X,* X. X,* X. X,* X. X,* X. X,* X. X,** #

UR: # b. b, a. a, h. h, b. b, a. a, h. h, #

RR: # *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, R, R, R, B. B, B, B, B, B, #

*CONTOUR: This constraint bans contour segments (essentially, bans the continuation version of a segment, on the surface). This is sort of an automatic constraint that is not part of the analysis; any generation process using this representation should include *CONTOUR just after the final IO-INTEGRITY or IO-UNIFORMITY constraint.

SR: # *X*. *X*. - *X*. *X*. - #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # *X*. *X*. - *X*. *X*. *X*. #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # *X*. *X*. *X*. *X*. *X*. - #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # *X*. *X*. *X*. *X*. *X*. *X*. #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #

IDENTIO([−CONS]): Keep vowels as vowels. Normally *X* aligned with a vowel would become *V*, but since this simplified language has only one vowel, the candidate set just includes that one.

SR: # *X*. a. - *X*. a. - #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # *X*. a. - *X*. a. *X*. #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # *X*. a. *X*. *X*. a. - #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # *X*. a. *X*. *X*. a. *X*. #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #

IDENTIO([+CONS]): Keep consonants as consonants. All remaining *X* instances have become *C*.

SR: # *C*. a. - *C*. a. - #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # *C*. a. - *C*. a. *C*. #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # *C*. a. *C*. *C*. a. - #

UR: # b. a. h. b. a. h. #

RR: # ? ? ? ? ? ? #

MR: # R. R, R, B. B, B, #,

SR: # *C*. a. *C*. *C*. a. *C*. #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #

IDENTIO([+SPREAD-GLOTTIS]): Now all *C*s aligned with 'h' have become 'h'.

SR: # *C*. a. - *C*. a. - #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # *C*. a. - *C*. a. h. #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # *C*. a. h. *C*. a. - #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # *C*. a. h. *C*. a. h. #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #

IDENTIO([-SPREAD-GLOTTIS]): The remaining *C* instances have become 'b'.

SR: # b. a. - b. a. - #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # b. a. - b. a. h. #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # b. a. h. b. a. - #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # b. a. h. b. a. h. #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #

*DoubleSequence([-cons],[+sglot],[-cons]): Due to the lack of a triggering environment, this constraint has no effect.

SR: # b. a. - b. a. - #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # b. a. - b. a. h. #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # b. a. h. b. a. - #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #,

SR: # b. a. h. b. a. h. #

UR: # b. a. h. b. a. h. #

RR: # *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, #

Introduce Reduplicant Reference Property MCFG: The candidate set is now intersected with the MCFG given earlier in (2.4). The resulting candidate set is represented as an MCFG and obeys the reduplicant reference property. The reduplicant reference tier contains a base-aligned copy of the reduplicant, ready for the next constraint.

SR: # b. a. - b. a. - #

UR: # b. a. h. b. a. h. #

RR: # - - - b. a. - #

MR: # R. R, R, B. B, B, #,

SR: # b. a. - b. a. h. #

UR: # b. a. h. b. a. h. #

RR: # - - - b. a. - #

MR: # R. R, R, B. B, B, #,

SR: # b. a. h. b. a. - #

UR: # b. a. h. b. a. h. #

RR: # - - - b. a. h. #

MR: # R. R, R, B. B, B, #,

SR: # b. a. h. b. a. h. #

UR: # b. a. h. b. a. h. #

RR: # - - - b. a. h. #

MR: # R. R, R, B. B, B, #

DepBR(C): The candidate [bahba] contains an element in the reduplicant that is not in the base, thus violating DepBR(C), and it is therefore eliminated.

SR: # b. a. - b. a. - #

UR: # b. a. h. b. a. h. #

RR: # - - - b. a. - #

MR: # R. R, R, B. B, B, #,

SR: # b. a. - b. a. h. #

UR: # b. a. h. b. a. h. #

RR: # - - - b. a. - #

MR: # R. R, R, B. B, B, #,

SR: # b. a. h. b. a. h. #

UR: # b. a. h. b. a. h. #

RR: # - - - b. a. h. #

MR: # R. R, R, B. B, B, #

MaxIO(C): DepBR(C) didn't really play any role in this generation process, as MaxIO(C) would have eliminated the same candidate. At any rate, it goes on to eliminate all the other h-deleting candidates, leaving the candidate which violated no constraints.

SR: # b. a. h. b. a. h. #

UR: # b. a. h. b. a. h. #

RR: # - - - b. a. h. #

MR: # R. R, R, B. B, B, #

The correct output, [bahbah], is derived.

2.5.2.2 *Suffixed Reduplication*

Now for an interesting example, in which an /h/ deleted in the stem duly fails to show up in the copy: /red+bah+a/ becomes [babaa] where *[bahbaa] might be expected. By this point it should be clear that the constraints up to *VhV produce a candidate set that is exactly like the underlying form except that candidates with /h/-deletion are allowed. I will begin with that sort of candidate set, to avoid repetition:

SR: # b. a. - b. a. - a. #

UR: # b. a. h. b. a. h. a. #

RR: # *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, - #,

SR: # b. a. h. b. a. - a. #

UR: # b. a. h. b. a. h. a. #

RR: # *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, - #,

SR: # b. a. - b. a. h. a. #

UR: # b. a. h. b. a. h. a. #

RR: # *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, B, #,

SR: # b. a. h. b. a. h. a. #

UR: # b. a. h. b. a. h. a. #

RR: # *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, - #

*DoubleSequence([-cons],[+sglot],[-cons]) [*VhV]: The final /h/ is intervocalic, so *VhV eliminates the candidates in which it is preserved.

SR: # b. a. - b. a. - a. #

UR: # b. a. h. b. a. h. a. #

RR: # *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, - #,

SR: # b. a. h. b. a. - a. #

UR: # b. a. h. b. a. h. a. #

RR: # *?* *?* *?* *?* *?* *?* *?* #

MR: # R. R, R, B. B, B, - #

Introduction of Reduplicant Reference: Here the reduplicant reference MCFG is intersected in, just as before.

SR: # b. a. - b. a. - a. #

UR: # b. a. h. b. a. h. a. #

RR: # - - - b. a. - - #

MR: # R. R, R, B. B, B, - #,

SR: # b. a. h. b. a. - a. #

UR: # b. a. h. b. a. h. a. #

RR: # - - - b. a. h. a. #

MR: # R. R, R, B. B, B, - #

DepBR(C): The candidate *[bahbaa] contains a consonant in the reduplicant that is not in the base, and is therefore eliminated, leaving the winning candidate.

SR: # b. a. - b. a. - a. #

UR: # b. a. h. b. a. h. a. #

RR: # - - - b. a. - - #

MR: # R. R, R, B. B, B, - #

MaxIO(C): Prior to this constraint the candidate set has only one candidate, so the constraint has no effect. The surviving candidate may be transcribed as [babaa]; it is the observed surface form.

2.5.3 A FURTHER EXAMPLE: MALAY NASAL SPREADING

McCarthy & Prince (1995) introduced Malay nasal spreading as a major argument for their theory, as the pattern is difficult to explain in a rule-based way. The data used comes from a single source, however, and has never been confirmed (see Raimy (2000)). The following makes no claims with respect to the accuracy of the data, but merely shows how McCarthy & Prince's (1995) analysis may be encoded. In Malay, nasality persists from a nasal consonant or vowel onto the following vowel or glide. Thus what might be underlying /waŋi/ ('fragrant') appears on the surface as [waŋ̃ĩ]. Interestingly, in total reduplication nasality from the final vowel of one copy appears to spread to the initial part of the other and from there is again copied to the first copy[5], resulting in [w̃ãŋ̃ĩw̃ãŋ̃ĩ] rather than *[waŋ̃ĩwaŋ̃ĩ] or *[waŋ̃ĩw̃ãŋ̃ĩ] as might be expected. The following derivation processes yield the facts just described. See McCarthy & Prince (1995) starting at page 45 for further details and discussion. Note that, due to space limitations, in what follows the candidate sets at each stage will be described rather than explicitly depicted.

[5] In Malay total reduplication it is impossible to distinguish a reduplicant and a base. In this example I have assumed that the reduplicant precedes the base, but the analysis works equally well in the other direction

2.5.3.1 *Simple Form*

To begin, the output of GEN as applied to /waŋi/ is the set of all surface strings with underlying /waŋi/ included, as usual. The three most highly-ranked constraints are IO-INTEGRITY(X), IO-UNIFORMITY(X), and *CONTOUR. Together these produce a candidate set in which coalescence and splitting are not allowed. All candidates are formed by insertion, deletion, or feature change. Next, DEPIO(X) removes the possibility of segment insertion, and MAXIO(X) removes the possibility of segment deletion. Now the candidates include all four-segment strings from the segment set of Malay.

Following this, there is a bank of IDENTIO constraints: IDENTIO([±SYLLABIC]), IDENTIO([±CONSONANTAL]), IDENTIO([±SPREAD-GLOTTIS]), IDENTIO([±HIGH]), IDENTIO([±BACK]), IDENTIO([±LOW]), IDENTIO([±LABIAL]), IDENTIO([±DORSAL]), and IDENTIO([±CORONAL]). After these constraints, the candidate set includes only variations of nasality: [waŋi], [waŋĩ], [wãŋi], [wãŋĩ], [w̃aŋi], [w̃aŋĩ], [w̃ãŋi], and [w̃ãŋĩ].

IDENTIOIN([±NASAL], [+CONS]): This constraint preserves nasality in consonantal segments (this does not include glides). It has no effect on the current candidate set.

*SEQUENCE([+NASAL], [−NASAL], [+SPREAD-GLOTTIS]): This constraint could be written *[+nasal]h_0[−nasal]. It essentially enforces nasal spreading, transparent to glottals. All remaining candidates that feature a nasal followed by a non-nasal are removed, leaving the following: [waŋĩ], [wãŋĩ], and [w̃ãŋĩ].

IDENTBR([±NASAL]): This constraint has no effect on the current candidate set, as it is not reduplicative.

*[+NASAL]: This constraint bans all nasal segments. The effect this has in the grammar is that underlying nasal marking only matters in consonantal (non-glide, non-vowel) segments. Therefore, the underlying form /w̃ãŋ̃i/ would have led to the same output as /waŋi/. The candidate set is now reduced to its least nasal member: [waŋ̃i].

IDENTIO([±NASAL]): Since this constraint, which preserves underlying nasality, is ranked below *[+NASAL], it has no effect regardless of the input.

The winning candidate is as observed, [waŋ̃i].

2.5.3.2 *Reduplicated Form*

The output of GEN as applied to /RED+waŋi/[6] is the set of all surface strings with underlying /waŋiwaŋi/ included, where the first half is marked as a reduplicant, the second half as a base. The top-ranked constraints, up to and including the bank of IDENTIO constraints, produce a candidate set where all candidates look like the underlying form except that any nasality pattern is possible, varying from [waŋiwaŋi] to [w̃ãŋ̃iw̃ãŋ̃i].

IDENTIOIN([±NASAL], [+CONS]): This constraint preserves nasality in consonantal segments (this does not include glides). It has no effect on the current candidate set.

[6]Once again, this representation assumes prefixing reduplication; the input could equally be /waŋi+RED/.

*Sequence([+nasal], [−nasal], [+spread-glottis]): This constraint could be written *[+nasal]h_0[−nasal]. It essentially enforces nasal spreading, transparent to glottals. All remaining candidates that feature a nasal followed by a non-nasal are removed, leaving the following: [waŋĩw̃ãŋĩ], [w̃ãŋĩw̃ãŋĩ], and [w̃ãŋĩw̃ãŋĩ].

IdentBR([+nasal]): This constraint requires that if a segment in the base is nasal, the corresponding segment in the reduplicant must be nasal as well, and vice versa. This reduces the candidate set to one: [w̃ãŋĩw̃ãŋĩ].

*[+nasal]: This constraint has no effect, as there is a single candidate.

IdentIO([±nasal]): This constraint has no effect.

The winning candidate is as observed, [w̃ãŋĩw̃ãŋĩ].

2.6 CONCLUSION

The preceding sections showed how to model the Base-Reduplicant Correspondence Theoretic framework of Optimality Theoretic Phonology by an extension of weighted finite state Optimality Theory. It also introduced a simple, efficient, and complete representation for phonological forms and a system for encoding Optimality Theoretic constraints as weighted finite state machines. The essential elements of this extension are the addition of a reference tier that duplicates the surface form of a candidate's reduplicant but aligns it with the base so that Base-Reduplicant correspondence relations can be computed via constraints that are essentially identical to Input-Output correspondence constraints. In

order to meet the reference identity requirement in candidates, the candidate set must be represented by a grammar with higher representational complexity than a finite state machine. Therefore the candidate set is represented by a multiple context free grammar (a type of mildly context sensitive grammar) and finite state intersection is replaced by weighted MCFG parsing.

There are practical consequences to this increase in complexity. In the original finite state method, each step in the winnowing process takes time proportional to the square of the number of states required to represent the candidate set being winnowed. Unfortunately, the size of the candidate set has the potential to increase exponentially as a result of repeated intersection (the intersection of an n-state FSM with an m-state FSM may have up to nm states, so in theory r m-state constraints applied to an n-state candidate set could lead to a nm^r-edge candidate set). The practical effect of this is that the efficiency of Optimality Theoretic generation is affected greatly by the ranking of constraints, and by the *size* (in other words, complexity) of the constraints. For example, a 1-state constraint can only reduce the size of a candidate set representation, so high ranking of such constraints makes for an efficient generation process. Essentially, an optimal ranking, from an efficiency point of view, consists of the available constraints ranked from the most simple (generally markedness constraints predicated on a single segment) to the most complex (for example, a faithfulness constraint with both left and right context).

The Base-Reduplicant Correspondence Theoretic extension using MCFGs makes the importance of rank order even greater. Each non-terminal symbol in a constraint set representation can become as many as n^4 after a constraint intersection, where n is the number of states in the constraint being intersected, so size increases faster, and the inter-

section itself is less efficient (on the order of n^6 instead of n^2 as before). The practical effect of this is that the conversion of the candidate set from an FSM to an MCFG, as discussed on page 71, must be delayed as long as possible, both to limit the number of intersections that must be made and to reduce the potential for combinatorial explosion. The other effect is that highly ranked constraints should be as simple as possible in terms of the number of states necessary to represent them (simple faithfulness constraints such as DEPIO(X), MAXIO(X), IO-UNIFORMITY(X), and IO-INTEGRITY(X) are especially important to rank highly). This is borne out by the common empirical observation that constraints are seldom needed that count past three, *i.e.,* ideally constraints should have no more than three states (in the compressed representation given earlier).

CHAPTER 3

An Analysis of Malagasy Phonology

3.1 INTRODUCTION

Optimality Theory offers a valuable framework for investigation of the phonological pattern of the world's languages, tying together as it does the formal description of a language's sound pattern with functional explanations for why that sound pattern falls out the way it does. It is, however, based on a complex computational model that makes hand-checking of analyses impossible, for all practical purposes. For this reason almost all analyses presented within the framework deal only with isolated phenomena, and only one or two have presented the full body of constraints that would be necessary to check whether the analysis in fact produces the actual surface forms of the language when given as input the hypothesized underlying forms[1]. Optimality Theoretic analyses have been focused primarily on questions of typology, looking at selected data patterns in multiple languages. As a result, I would assert, phonological theories based in the framework have missed out on the insights that might be gathered from in-depth analyses of individual languages. In fact the typological claims upon which most work in Optimality Theory has been focused may be suspect, based as they are on shallow analyses.

[1] One fairly complete analysis is McCarthy & Prince's (1993) analysis of Axininca Campa.

Given a computational model of Optimality Theory, as discussed in the previous chapter, it is possible to encode an analysis on a computer. Such analyses can be scaled up considerably while maintaining the same or a greater level of confidence in their correctness. The following sections present a trial of this analytical method.

The case study presented here is the Merina dialect (the standard dialect) of Malagasy, a West Austronesian language spoken by approximately 12 million people in Madagascar. The analysis given here is based upon a database of 509 Malagasy forms, each chosen to illustrate some aspect of the sound pattern of Malagasy, plus 192 constructed forms chosen to test the system's ability to rule out illegal forms, *i.e.* the completeness of the system's phonotactic component. The analysis attempts to cover the entire phonology of the language as it is currently known, and it is given in its entirety, with the complete set of constraints needed to check it, as well as a fixed ranking of those constraints. It has in fact been checked by computer against the aforementioned body of data, using my OTPad toolkit[2].

I have relied for data and generalizations primarily on Erwin (1995), Paul (1995), Hollanger (1973), and Keenan & Polinsky (1998). Keenan & Polinsky (1998) provide references for much of the Malagasy literature.

The analysis leads to a few conclusions of a typological or theoretical nature, one being a calling into question of the Base-Reduplicant Correspondence Theory (McCarthy & Prince 1995) analysis of reduplication as a universal analysis for all or most instances of reduplication. Although at first glance the reduplicative system of Malagasy seems to fit

[2]For a free copy of this software, which currently runs on Windows and UNIX platforms, please send a request to <albro@alum.mit.edu>.

with a Base-Reduplicant Correspondence Theoretic analysis, a more in-depth look at the data reveals that an analysis based on a simpler compounding-type model (the Morpheme Doubling model of Inkelas & Zoll (2000)) accounts for the data much more simply and completely than a Base-Reduplicant Correspondence Theoretic analysis could.

3.2 NECESSARY CHARACTERISTICS OF THE ANALYSIS

3.2.1 MORPHOLOGICAL LEVELS

I have analyzed Malagasy in the framework of Lexical Phonology and Morphology in Optimality Theory (LPM-OT; Kiparsky 2000). A comparison of this framework with other treatments of opacity may be found in §3.12.4.3. An LPM-OT analysis such as this one consists of three separate banks of constraints (although the same constraints will, at least theoretically, be present in each of the three banks—only the ranking will differ). The first bank of constraints will be referred to as the *stem grammar*, the second as the *word grammar*, and the third as the *post-lexical grammar.* For Malagasy the following generation algorithm was sufficient; further cross-linguistic study will doubtless produce some refinement:

1. The root, together with any stem-level affixes, is passed through the stem grammar.
2. The word-level affixes are added to the result of the previous step and the output of the concatenation is passed through the word grammar.
3. The result of the previous step is passed through the post-lexical grammar.

When I say a form is "passed through" a grammar, I mean that the Optimality Theoretic Eval specified for that grammar is applied to the output of Gen for that form. Note that compounding is produced here by having two or more forms pass independently through one level and having the outputs joined together as input to the next level, so compounds joined prior to word-level may exist, as well as compounds joined prior to the post-lexical level.

In the sections to follow, constraints described as undominated should be interpreted as being undominated in all three grammars, most crucially in the post-lexical grammar.

3.2.1.1 LPM-OT *Typographical Notes*

A complication that arises when using LPM-OT is that in addition to the two types of representations necessary with the standard model, *i.e.* the underlying form and the surface form, there are the output of the stem grammar and the output of the word grammar. In this document a form such as /manɣatah/ represents an underlying form, [maŋˈɣatah]$_s$ represents the output of the stem grammar, [manˈgatak]$_w$ represents the output of the word grammar, and something like [maŋˈgataka] represents the surface form of an utterance. Candidates in tableaux are not explicity labeled, but they are always the output of the next higher grammar from the source of the tableau's input form. For example, if the input form of the tableau is marked as a stem-level output (*e.g.,* [maŋˈgatah]$_s$), it may be assumed that the candidates are word-level outputs. Forms in data tables are always surface outputs unless otherwise marked (as in the case of hypothesized underlying forms).

In some forms I have indicated morpheme boundaries. In glosses and underlying

forms (in other words, in any place where the morpheme boundary is an explicit artifact of the analysis), I have used the symbol "+" to denote such a boundary. In data tables I have occasionally marked morpheme boundaries as well. Here they are included only as an aid to the reader (the boundary erasure convention of LPM-OT has removed them), and are indicated by the symbol "-" in order to distinguish them from boundaries that are active in the representation and analysis.

The final typographical note is that constraint rankings are generally specific to particular levels of grammar, so they are marked as such: "$\gg_s$" for a stem-level ranking, "$\gg_w$" for word-level ranking, and "$\gg_p$" for a post-lexical ranking.

3.2.2 REPRESENTATIONS

In order to explain some of the constraint families used in this analysis, it is necessary to briefly discuss the representational scheme that was used to build and check it. The representation was designed to be as simple as possible while still allowing analysis to succeed. The scheme is not particularly autosegmental (in the sense of Goldsmith (1976)); rather, the fundamental building block of a representation here is a phone. That is, to describe a given language in this scheme, one begins with a set of permissible surface phones. Features are then defined as natural classes, that is, sets of phones. These basic natural classes, defined by listing, may be added to by the application of set complement

(like -FEAT in standard terms) and intersection (like presence in a matrix, *e.g.,*

$$\begin{bmatrix} \text{+SON} \\ \text{-SON} \\ \text{+NAS} \end{bmatrix},$$

in standard terms). See §3.3.1 for a discussion of the features employed in this analysis.

3.2.2.1 *Representing Correspondence*

The computational model employed here does not mandate a strictly segmental representation, but it does tend to favor such a model, because the complexity of the generation system is exponential in the number of tiers used. I have found it useful here to adopt a conservative position with respect to hierarchical structure. I posit some structure, but nothing higher than complex segments. The term *complex segment* here encompasses simple segments such as *b* and *e,* and also bipartite segments such as diphthongs and affricates. In a language with a more complex syllable structure, it might be necessary to extend the hierarchy further, but for Malagasy this appears to be sufficient. Candidates in this scheme are represented as simple strings made up of the candidate surface form, the underlying form, and the correspondences between the members of both. A simplified example might be surface [pe] corresponding to underlying /ba/. This candidate could be represented as "b:p a:e." Here correspondence is indicated by the ":" character before which we place the underlying correspondent, and after which we place the surface correspondent. This representation allows a simple encoding of all the correspondences needed for this example, but not all the types of correspondence embodied in the full Correspondence Theory of McCarthy & Prince (1995). To be specific, it cannot encode correspondences involving

multiple segments (for example if a single surface segment corresponds to multiple underlying segments) or correspondences where the order has changed (metathesis). Both of these additional types of correspondences may be encoded in such a way as to allow more or less efficient implementation, but for this analysis only the many-to-one type is needed.

3.2.2.2 *Representing Complex Segments*

To allow for many-to-one correspondence, the representation of a segment is modified such that it can stretch. If surface [pe] were instead to correspond to underlying /pa͡i/ (as in $p_1a_2i_2$:p_1e_2, to use the usual correspondence-theoretic indexing notation), a first try at a representation might be "p:p a:e i:e." However, in this representation it is not clear that the two surface *e*'s in fact refer to the same segment. To make this distinction, I introduce segment boundaries: "|:| p:p |:| a:e |:e i:e |:|." This is a somewhat complicated representation, but it becomes clear in a more vertical format[3]:

UR:	|	p	|	a	|	i	|
SR:	|	p	|	e	e	e	|

As mentioned above, the boundaries here mark off complex segments; "eee" in the surface representation represents a single [e] segment and does not imply anything about phonetic duration ("e|e" *would* imply a longer duration). A complex segment that has multiple phones, such as an affricate or a diphthong, will be termed a *contour segment.*

[3] In the more compact representation of Chapter 2 this would appear as

UR:	p.	a.	i.
SR:	p.	e.	e,

Thus there are two types of sequences in the representation*:* *contour sequences* within a single complex segment, and *segmental sequences* which include multiple segments. The first type of sequence is dealt with by a family of *contour* constraints that operate within segments, such as *Contour([αplace], [−αplace]), which bans multiple places of articulation within a single segment. The second type is dealt with by *sequence* constraints that operate between segments, such as *Sequence([αplace, C], [−αplace, C]), which bans multiple places of articulation within a consonant cluster. See §3.2.3 for more information on these constraint families.

3.2.2.3 *Representing Non-Correspondence*

There is still an essential element missing from this representation of correspondence, however—non-correspondence. The representation encodes an area where something corresponds to nothing by means of the symbol –. In the following representation

UR:	|	m	|	b	|	a	|	t	|
SR:	|	p	p	p	|	e	|	–	–

the underlying consonant /t/ has no surface correspondent, *i.e.* $m_1b_1a_2t_3$:p_1e_2.

3.2.2.4 *Summary*

To summarize, the representation uses segment boundaries and segment interior symbols to establish one-to-one or one-to-many correspondence. It cannot represent correspondence in cases of metathesis[4]. The symbol "–" in a representation indicates non-

[4]The representation can be extended for metathesis as well, as laid out in Appendix F.

correspondence.

3.2.3 THE CONSTRAINT COMPONENT

3.2.3.1 *Difference from Standard Correspondence Theory*

The constraints used in this analysis are members of a restricted set of parametrized constraint families more or less approximating the family of constraints assumed in McCarthy & Prince (1995). They differ from standard Correspondence Theory constraints in the following ways:

1. They use the definition of correspondence from §3.2.2 rather than the standard one.

2. They use a slightly different model of contextual faithfulness. The conventional model (Steriade 1998; Steriade 2000 (in press); Beckman 1998) uses surface contexts for contextual IDENTIO, but here contextual IDENTIO uses underlying contexts, an underexplored alternative. There are four reasons for maintaining this difference here: (1) in Malagasy there are contexts that appear identical on the surface but differ in terms of their faithfulness based on a context difference underlyingly; (2) it is possible to use underlying contexts with this representation because they are not syllable-bound (that is, surface contextual faithfulness is often based on syllable positions, which under the concept of Richness of the Base cannot be guaranteed to to exist in underlying forms, whereas these contextual faithfulness constraints are based on segments or morphological edges); (3) surface and underlying contexts are equivalent in terms of their computational complexity (the constraint automata

used for one or the other use an equivalent number of states), but the analysis given here shows that underlying contextual faithfulness can account for opaque phenomena that are unamenable to explanation via surface contextual faithfulness; and (4) the intuition is that certain positions in the reference form (UR or Base) are more salient—easier to remember by the speaker, more likely for change to be noticed by the listener. With that said, I am not claiming, pending further investigation, that there is no place for surface-contextual IdentIO; it simply has not been necessary for Malagasy.

3. They include a few constraint families that could be termed "two-level constraints." Such families are essentially markedness constraints that have the unusual ability to refer to the underlying form. They originate with pre-Optimality Theory work by Koskenniemi (1983); see Kager (1999) for an introduction to their use in an Optimality Theoretic context. These two-level constraints can be used to account for opacity, among other things. In this analysis, however, I have avoided the use of two-level constraints where possible; their only use is the constraint OpqStress introduced in §3.8.1. There may be legitimate uses of two-level constraints in accounting for opacity, but in general their use is stipulative rather than explanatory, and loses the typological generality that can be characteristic of Optimality Theoretic analyses.

4. They are uniformly implementable as weighted finite state machines, which rules out certain overly powerful constraint families (see, for example, Eisner (1997d)).

Most of the constraint families used in this analysis are fairly standard families taken from the Optimality Theoretic literature, but some of them are new, and some of them have slight but important differences from the standard constraint families of the same name. Thus, it will be worthwhile to spend some time reviewing the details of each before heading into the analysis.

A particular constraint is formed from a constraint family by supplying one or more parameters. These parameters are, in general, the names of natural classes; *i.e,* they represent sets of segments. Throughout the analysis I have typically named these natural classes in the traditional way, using a feature matrix. Note that in examples illustrating particular members of the constraint families, I have used features from the feature set defined in §3.3.1.

3.2.3.2 *Faithfulness Constraints*

Most of the faithfulness constraints are the standard Correspondence constraints of McCarthy & Prince (1995), but as mentioned above there are contextual versions of many of the families, and some families vary slightly from the standard. Note in the constraint definitions to follow that S refers to a natural class (feature matrix) matched against the surface representation, U refers to a class matched against the underlying form, B refers to one that is matched against both, and M refers to a morphological context. Further note that where a family is listed as *Name*IO (that is, input-to-output faithfulness), there are also versions of the family with input-to-reduplicant (IR), input-to-base (IB), and base-to-reduplicant (BR) faithfulness.

MaxIO(ʊ) For each segment of natural class ʊ in the UR, there must exist at least one corresponding segment in the SR. Output a violation for each underlying segment of natural class ʊ for which there is no corresponding surface segment. There are positional variants, *e.g.*, MaxIO($ʊ_1$)/____$ʊ_2$, which acts like MaxIO($ʊ_1$) except that the trigger segment must be followed in the underlying form by a segment matching $ʊ_2$ for any violation to accrue.

DepIO(s) For each segment of natural class s in the SR, there must exist at least one corresponding segment in the UR. Output a violation for each surface segment matching s for which there is no corresponding underlying segment. As with MaxIO, the DepIO family includes positional variants. For example, DepIO(s)/____ʊ acts like DepIO(s) except that the trigger segment must be followed in the underlying form by a segment of natural class ʊ for any violation to accrue.

IO-Integrity(ʊ) A family of constraints that disallows splitting. Its operation is as follows: a segment of natural class ʊ in the underlying representation (UR) may not correspond to more than one segment in the surface representation (SR). Output a violation for each surface segment beyond the first one corresponding to an underlying segment matching ʊ. In representational terms, this constraint outputs a violation for each instance in a candidate of "ʊ:|"; that is, of the surface form having a segment divider aligned with an underlying member of natural class ʊ. It is the opposite of IO-Uniformity, *q.v.*

IO-UNIFORMITY(S) A ban on coalescence, which operates as follows: a segment of natural class S in the SR may not correspond to more than one UR segment. Output a violation for each underlying segment beyond the first one corresponding to a surface segment matching S. For example, if S were "C" (equivalently, "[-SYL]") this would output a violation for the candidate

UR:	a	n	t	a		
SR:	a	n	n	n	a	

.

FEATMAXIO(B) (*feature max*) If a segment of natural class B appears in the underlying form, then a segment of class B must appear on the surface somewhere aligned with it. Outputs a violation for each such segment appearing in the underlying form without a vertically aligned matching correspondent on the surface. Note that this is not a standard family; it is essentially the local conjunction of MAXIO(B) with PTIDENTIO(B) (see below). The family has all of the positional variants that MAXIO has.

IDENTIO(B) Given corresponding segments in the UR and the SR, if the underlying segment is a member of natural class B then the surface segment must be a member of B as well. Output a violation for each underlying member of B that corresponds to a surface segment that is not in B. In addition, the family may be specified as IDENTIO(±B) for a bidirectional constraint more similar to the standard IDENTIO family of McCarthy & Prince (1995) (this is like IDENTIO(B) except that additionally if a surface segment matches B then its underlying correspondent, if any, must match B as well). Positional variants exist as well. For example, IDENTIO(B)/____U

acts like IDENTIO(B) except that the trigger segment must be followed on the underlying form by a segment matching U for any violation to accrue (see §3.2.3.1 for more discussion of the somewhat non-standard model of contextual faithfulness used here). The version of IDENTIO given here is both more and less powerful than the standard. First, the way in which this IDENTIO is more powerful: the major difference between IDENTIO here and in the standard treatment is that the parameter, B, is not a feature, but a *natural class*, so the standard IDENTIO([±CONT]) constraint would here be replaced by the equivalent IDENTIO(±[+CONT]), and a constraint like IDENTIO(±[+CONT,-SON]) is possible—this constraint would penalize a change from a fricative to something that is not a fricative, or vice versa. (Note that this extension can be modeled (albeit less efficiently) by the IDENTIOIN family – IDENTIO(±[+CONT,-SON]) is equivalent to IDENTIOIN([+CONT], [-SON]) ranked together with IDENTIOIN([-SON], [+CONT]).) IDENTIO as given here is less powerful than the standard version just in one case—the standard IDENTIO constraint family is often given without parameters, with the meaning "penalize any corresponding segments if they are different in any particular," whereas no single constraint in the IDENTIO family given here can have that meaning (such a constraint would be more complicated computationally, requiring additional states).

PTIDENTIO(B) (Input-Output Identity at a Point) This is a non-standard constraint family that works like IDENTIO(B) except that where IDENTIO(B) outputs a violation if two corresponding segments differ at any point (say an underlying segment corresponds to a surface contour), PTIDENTIO(B) outputs a violation only if at no point are the two segments identical with respect to their membership in natural class B.

For example, the candidate

UR:	|	t	|	a a	|
SR:	|	t	|	a u	|

violates IDENTIO([+LOW]), but not PTIDENTIO([+LOW]) because the corresponding segments are both [+LOW] at one point.

PRESERVE{RT,LFT}(U, M) (A family of constraints that takes the place of ANCHOR{RT,LFT} plus CONTIGUITY.) PRESERVERT outputs a violation for each underlying segment of natural class U, within a morphological context M, that does not have a surface correspondent, but only after the first such underlying segment that does have a surface correspondent. For example, compare the following candidates[5] with respect to PRESERVERT(X, RED)[6]:

M:	-	-	-	-	-	-	-	-	red	red	red	red	red	red	red	red	red		
UR:	|	v	|	u	|	v	|	u	|	v	|	u	|	v	|	u	|
SR:	|	v	|	u	|	v	|	u	|	v	|	-	-	-	|	u	|

and

M:	-	-	-	-	-	-	-	-	red	red	red	red	red	red	red	red	red		
UR:	|	v	|	u	|	v	|	u	|	v	|	u	|	v	|	u	|
SR:	|	v	|	u	|	v	|	u	|	-	-	-	|	v	|	u	|

In the first candidate, the first segment of the reduplicant has a surface correspondent but the next two segments do not, so PRESERVERT(X, RED) is violated twice.

[5] Note — this is the Malagasy form for *to bark (*[vu'vu], reduplicated as [vu,vu'vu]).

[6] The parameters here indicate that the morphological context is the reduplicant—the morphological context parameter specifies the contents of the morphological tier where the constraint is to be valid. The "X" indicates that all segments are being preserved.

In the second candidate, the first two segments of the reduplicant have no surface correspondents, but this is ignored since PRESERVERT starts counting violations after the first faithful segment; thus, the second candidate does not violate PRESERVERT(X, RED). The mirror image constraint, PRESERVELFT, outputs a violation for each deleted segment preceding the last preserved segment in the morphological context. For PRESERVELFT, the equivalent candidates to those given above are

M:	red	red	red	red	red	red	red	red	red	-	-	-	-	-	-	-	-
UR:	|	v	|	u	|	v	|	u	|	v	|	u	|	v	|	u	|
SR:	|	v	|	-	-	-	|	u	|	v	|	u	|	v	|	u	|

and

M:	red	red	red	red	red	red	red	red	red	-	-	-	-	-	-	-	-	
UR:	|	v	|	u	|	v	|	u	|	v	|	u	|	v	|	u	| ,
SR:	|	v	|	u	|	-	-	-	|	v	|	u	|	v	|	u	|

where the first candidate incurs two violations of PRESERVELFT(X, RED) and the second incurs none, since in the first candidate the last faithful segment, the one that terminates the violation counting, is the final /u/ of the reduplicant, whereas in the second candidate it is the first /u/. Note that candidate

M:	red	red	red	red	red	red	red	red	red	-	-	-	-	-	-	-	-	
UR:	|	v	|	u	|	v	|	u	|	v	|	u	|	v	|	u	|
SR:	-	-	-	-	-	-	-	-	|	v	|	u	|	v	|	u	|

would incur no violations of the PRESERVELFT(X, RED) constraint.

3.2.3.3 *Markedness Constraints*

Most of the standard markedness constraints found in the literature should be representable as some member of one of the following families, unless they rely upon some complex structural property of the representation. In addition to the variables found in the previous section, I have added the variable I here (for "intervening ignored surface segments") to indicate a natural class of segments that are ignored.

*(S) Bans segments of natural class S from the SR. Outputs a violation for each surface segment from natural class S. For example, [azusa] violates *([+CONT]) five times and *([+CONT,-SON]) twice. An extended version of this family exists — *M(S, M) — where the violation is triggered only for segments within a specified set of morphological domains. For this analysis the morphological domains are limited to "inside a reduplicant," "inside a base of reduplication," and "inside neither a base nor a reduplicant."

*INITIAL(S, I) Bans S in SR if preceded only by members of I — *(#(|I*)+S). For example, *INITIAL([-LOW], C) would output a violation for a words such as [ɛŋθ] or [stɪɛŋθ] in which the first vowel of the word is non-low, whereas *INITIAL([-LOW], ∅)[7] would output a violation for [ɛŋθ] but not for [stɪɛŋθ]. There is a morpheme-bound variant *INITIALM(S, I, M) that outputs a violation if the first post-I element in morpheme M is of natural class S. For example, *INITIALM([-PRIM], C, RED) requires the initial vowel of a reduplicant to carry primary stress. Caveat: a *INITIAL(S,I) constraint is only valid if natural classes S and I do not intersect. For sake of brevity, in cases

[7] ∅ is the empty natural class—no segment is a member of it.

where the I parameter of a constraint is set to ∅, I will generally omit the parameter; thus *Initial([-cont]) is equivalent to *Initial([-cont], ∅).

*Final(S, I) This is the mirror image of *Initial—Bans S in SR if followed only by members of I — *(S(I*|)$^+$#).

*Contour(S_1, S_2, I) Bans members of natural class S_1 from following members of class S_2 *within* a segment (*i.e.,* within the segment boundary characters "|"). Zero or more members of natural class I may intervene—*(S_1I*S_2). For example, *Contour([+high], [+low], [-high,-low]) would ban an [i͡a] diphthong or a [ɪ͡ea] triphthong (weird as such a thing might be). It is required that natural class I intersect with neither S_1 nor S_2. If I is the empty set (∅), then the segments S_1 and S_2 must be adjacent for violations to accrue; thus, *Contour([+high], [+low], ∅)[8] would ban [i͡a] but not [ɪ͡ea]. Since *Contour only applies within a single complex segment, *Contour([+high], [+low], ∅) would not be violated by [ia].

*Sequence(S_1, S_2, I) This is like *Contour except that it operates between segments. That is, for such a constraint to be violated, there must be a segment boundary intervening between the members of classes S_1 and S_2—*(S_1(I*|)$^+$$S_2$). For example, *Sequence([αrnd, V], [-αrnd, V], C)[9] would enforce roundness harmony in vowels, where intervening consonants are transparent to the harmony.

*InitialSequence(S_1, S_2, I) Like *Sequence, except that it only notices sequences at the beginning of words — *#(I*|)$^+$$S_1$(I*|)$^+$$S_2$. For example, *InitialSequence([V,-

[8] Typically written *Contour([+high], [+low]).

[9] Note that this is actually an abbreviation for the constraint sequence *Sequence([+rnd, V], [-rnd, V], C), *Sequence([-rnd, V], [+rnd, V], C).

stress], [V,-stress], C) penalizes a stress lapse at the beginning of a word (the mirror image of this, *FinalSequence([V,-stress], [V,-stress], C) is employed in the Malagasy analysis).

*FinalSequence(S_1, S_2, I) This is the equivalent of *InitialSequence, but at the end of the word.

Exists(S) Outputs one violation for each word in which no surface segment matches S. For example, Exists([+stress]) would output a single violation for a word with no stress.

More on *Sequence and *Contour The use of constraint families such as these, which allow the banned sequence to be non-local in the sense that they can skip over zero or more intermediate segments (*e.g.,* *'VC_0'V or in the terms used here *Sequence([V, +stress], [V, +stress], C)—a ban on stressed vowels separated by zero or more consonants), is somewhat controversial. A common restriction in current phonological work is that *Sequence-type constraints may only refer to elements which are adjacent in the representation in some way. The intent of this restriction is to reduce the formal power of constraints, but the effect of the restriction in actual practice is to complicate representations sufficiently to allow non-adjacent segments to fall under the purview of such constraints. The additional formal power gained by this complication of the representation in fact outweighs the formal power lost by accepting the strict adjacency condition on constraints. In terms of the computations involved in a finite-state model of Optimality Theory (Ellison 1994b; Eisner 1997c; Albro 1998a; Frank & Satta 1998; Karttunen 1998;

Eisner 2000)[10], the complexity of a representation might be defined by the number of states necessary to represent it as a finite state machine, and similarly the complexity/power of a constraint might be defined in terms of the number of states necessary to represent it. In the case of *SEQUENCE([V,+STRESS], [V,+STRESS], C), the addition of an ignored class of intermediate segments requires only the addition of a single arc to the finite state machine that represents the constraint, whereas an analysis of stress without such a constraint would require the addition of explicit syllables in the representation. Such an addition almost doubles the number of states in a candidate set representation[11]. There is one necessary restriction on the *SEQUENCE constraints, however — in a constraint matching the template *SEQUENCE(S_1, S_2, I) the natural class I (the parameters of a constraint class are always natural classes, which may be thought of as sets of phonemes) may not intersect with the natural classes S_1 or S_2. There is no reason that the set of allowable *SEQUENCE parameters could not be restricted further.

Aside from computational considerations, there are also empirical arguments against a strict adjacency requirement for *SEQUENCE constraints. See, for example, Frisch *et al.* (in press) and Ringen & Heinämäki (1999).

[10]Amongst these references Ellison 1994b is the original article on the weighted finite state model of Optimality Theory, and Frank & Satta 1998 plus Karttunen 1998 are the original articles on the transducer model. Eisner 1997c, Albro 1998a and this dissertation expand on the weighted finite state model (Eisner 1997c mostly consists of a new representation and constraint formalism, and Albro 1998a modifies this representation and the formalism to deal with certain problems with the representation of non-correspondence; this dissertation expands the weighted finite state model to areas where GEN is not finite state and provides yet another new representation and constraint formalism).

[11]Of course an explicit syllable would also be useful for modeling a weight-based stress system, but one might still do without explicit syllables by treating weight as a feature (*i.e.,* adding heavy versions of each vowel phone).

3.2.3.4 *Two-Level Constraints*

The following constraint families can be used to construct fairly arbitrary constraints, of which some would be considered two-level.

*SequenceIO(S_1, U_1, S_2, U_2, I) S_1 corresponding to U_1 may not precede S_2 corresponding to U_2 with only I intervening — $*(S_1 : U_1 I^* (|I^*)^* S_2 : U_2)$. This has the usual IR, IB, and BR variants.

*ContourIO(S_1, U_1, S_2, U_2, I) Similar to *SequenceIO, but confined to the interior of a segment. This has the usual IR, IB, and BR variants.

3.2.4 BACKGROUND ON MALAGASY

A complete description of Malagasy morphology and syntax is beyond the scope of this endeavor. Please refer to Keenan & Polinsky (1998) and works cited therein for further details. For the purpose of this document, it should be sufficient to list the affixes that appear in the data together with the abbreviations that are used in glosses.

Abbrev.	Expansion	Note/Examples
pres.	*present*	Prefixed *m-* appears in pres. tense and imperative forms (word-level prefix)
imp.	*imperative*	The imperative suffix. *-a* in the examples used here (word-level)
act.	*active*	Active voice marker, differs from one verb to another. Allomorphs: *aN-, a-, i-, ∅–, aha-* (stem-level prefix)
pass.	*passive*	Passive voice marker, differs from verb to verb. *-an, -in* in the examples used here; there are also prefixes.
red.	*reduplication*	Usually has a diminutive or pejorative meaning.
circum.	*circumstantial*	Used when oblique obj. or adjunct of a vb. is made the subject. The suffix is *-an,* a word-level suffix.
gen.	*genitive*	Genitive marker; word-level, placed between possessed object and possessor.

3.3 SEGMENT INVENTORY

3.3.1 FEATURE SYSTEM

The representation used here represents phones directly, based on the assumption that native speakers have determined the allophones of their language and used this as the basis for learning a grammar. The feature system, then, is based on natural classes, that is, sets of phones. In addition to the surface allophones, the analysis below makes use of a

few abstract (that is, non-surface) phones—"β," "ɸ," and "ɣ"—which may be defined by the features already necessary to describe the surface phones.

The following features are used in this analysis (for values see charts below in SS3.3.2 and 3.3.3):

SYL Syllabic. [+SYL] is abbreviated as V, [-SYL] as C.

NAS Nasal (articulated with raised velum).

LAB Labial articulator.

COR Coronal articulator.

DORS Dorsal articulator.

ANT Relevant only to coronals, indicates point of articulation at or in front of the alveolar ridge.

SON Sonorant.

VCE Voiced.

CONT Continuant.

SGLOT Spread glottis.

HIGH High, applies to vowels.

LOW Low, applies to vowels.

BACK Back, applies to vowels.

ROUND Rounded vowel.

LAT Lateral airflow.

STRID Strident (increased turbulence in a fricative), applies here to s, z, ʧ, f, and v.

STRESS Stressed (of a vowel).

PRIM Having primary stress (of a vowel).

X Lexical flag for the opaque prefix *an-;* an arbitrary lexical marking—see §3.7.2.2.

3.3.2 CONSONANTS

The following are the consonantal segments of Malagasy, given in IPA transcription:

(1)

			labial	coronal		dorsal	glottal
				alveolar	post-alveolar		
obstruent	stop	-voice	p	t,ts	ʧ	k	
		+voice	b	d,ʤ,d͡r		g	
	cont	-voice	f	s			h
		+voice	v	z			
sonorant	cont	+nasal	m	n	n̲	ŋ	
		-nasal		l,ɾ			

Malagasy orthography represents ʤ as "j," ʧ as "tr," and d͡r as "dr." Otherwise, the orthography and the transcription agree. Note that this consonant inventory is based largely

upon the inventory given in Keenan & Polinsky (1998) but differs from it by not including prenasalized stop consonants, which are treated here as sequences.

The consonants I've described as palato-alveolar may in fact be alveolo-palatal; phonetic study is needed. In general they are further back than the alveolars, and are laminal rather than apical. Note that in the sections to follow the segment /ʧ/ is occasionally split into /t̲/ plus /ʃ/, mainly to describe constraints that forbid such a split (that is, /ʧ/ is a complex segment made up of the phones /t̲/ plus /ʃ/, which do not appear by themselves). The /ɾ/ is an alveolar tap, and /d͡ɾ/ is a pre-stopped r of some kind (Ying Lin, p.c.). The /h/ is often written but not pronounced; when pronounced it is glottal.

In terms of the feature system used in this analysis, the consonants are analyzed as follows:

	p	b	f	v	ɸ	β	m	t	d	s	z	n	l	ɾ	t̲	ʃ	k	g	ɣ	ŋ	h
SYL	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
NAS	-	-	-	-	-	-	+	-	-	-	-	+	-	-	-	-	-	-	-	+	-
LAB	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COR	-	-	-	-	-	-	-	+	+	+	+	+	+	+	+	+	-	-	-	-	-
DORS	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	+	-
SGLOT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+
ANT	-	-	-	-	-	-	-	+	+	+	+	+	+	+	-	-	-	-	-	-	-
SON	-	-	-	-	-	-	+	-	-	-	-	+	+	+	-	-	-	-	-	+	-
VCE	-	+	-	+	-	+	+	-	+	-	+	+	+	+	-	-	-	+	+	+	-
CONT	-	-	+	+	+	+	-	-	-	+	+	-	+	+	-	+	-	-	+	-	+
HIGH	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LOW	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
BACK	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ROUND	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LAT	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-	-	-	-
STRID	-	-	+	+	-	-	-	-	-	+	+	-	-	-	+	+	-	-	-	-	-

3.3.2.1 *Consonant Contour Inventory Constraints*

The inventory of segments allowable in candidates is fixed by listing[12], but the way these combine into segment positions (the space between "|" symbols in the representation) needs to be specified by constraint. The following undominated constraints determine the allowed affricates of Malagasy.

(2) *CONTOUR([+SON], [-SON]):

No segment contains a sonorant component followed by an obstruent one. Short form: *|[+SON] [−SON]|.

(3) *CONTOUR([αPLACE, C], [-αPLACE, C])

Segments are pronounced in only a single point of articulation. Short form: *|[αPL, C] [−αPL, C]| Note that this is actually an abbreviation for a fairly large number of constraints: *CONTOUR([−ANT], [+ANT]), *CONTOUR([+ANT], [-ANT]), *CONTOUR([LAB], [DORS]), *CONTOUR([DORS], [LAB]), *CONTOUR([COR], [DORS]), etc. The place features for consonants are LAB, DORS, COR, and ANT.

(4) *CONTOUR([αLAT], [−αLAT]):

There are no consonant contours involving [l]; i.e., *[l] only appears as a simple segment.* This constraint bans, for example, [d͡l]. Short form: *|[αLAT] [−αLAT]|

[12]This is not inevitable—the computational model would allow for a system where all permutations of the base features are included as possible segments and simple constraints ban illegal feature combinations. However, as stated before, it was assumed here that before learning an Optimality Theoretic grammar for a language, each child has already discovered the set of allophones for their language and would therefore use that as the candidate inventory. The system in which illegal segments are ruled out by constraint would be slightly less efficient computationally.

(5) *Contour([-cont], [+cont, -cor]):

All affricates are coronal. Short form: *|[-cont][+cont,-cor]|. Note that I could have mentioned [-cor] in both parts of this, but it was unnecessary due to the homorganicity constraint above.

(6) *Contour([αnas], [−αnas]):

No segment has a nasal part and a non-nasal part. Short form: *|[αnas][−αnas]|.

(7) *Contour([+cont], [-cont]):

No continuant-stop contours are allowed within a consonantal complex segment (since consonantal complex segments in Malagasy act as what would elsewhere be termed as syllable onsets, and a continuant-stop contour would constitute a decrease in sonority heading into a syllable). Short form: *|[+cont][-cont]|.

(8) *Contour([αvce], [-αvce]):

No segment may have a voicing contour. Short form: *|[αvce][-αvce]|

These constraints are all *Contour constraints because in this implementation the inventory of phones is given explicitly; no phone will be considered in a candidate outside of the list given in the table above. The same consideration will apply when specifying vowel constraints. There are also a few non-contour constraints which mandate that post-alveolars must appear only in contours[13]:

[13]These constraints look like the classic problematic spe rules that diagnose a need for explicit syllables (McCawley 1974). Here, though, they are simply diagnosing a possible need to add constraints that allow more fine-grained control of complex segments, such as a requirement that some phone not appear at the left or right edge of a complex segment. These constraints would then be *SegmentInitial(ʃ) and

(9) *Sequence(t̲, X):

t̲ may not directly precede another segment. Short form: *t̲X.

(10) *Final(t̲):

[t̲] may not end a word. Short form: *t̲#.

(11) *Sequence(X, ʃ):

The phone ʃ may not appear directly after any segment. Short form: *Xʃ

(12) *Initial(ʃ):

The phone ʃ may not begin a word. Short form: *#ʃ.

3.3.3 vowels

Malagasy has a four vowel system, consisting of the vowels /a/, /e/, /i/, and /u/. The /i/ vowel appears standardly as "y" in the orthography when word-final, and /u/ is standardly written "o." There are vowel diphthongs: /a͡u/, /a͡i/, and /u͡i/.

*SegmentFinal(t̲). Alternatively, there could be a family *Simple(S) that prevents segments of natural class S from appearing in anything except for a multi-phone contour.

The natural classes into which the vowels are divided are expressed by the following feature matrix:

	a	i	e	u
SYL	+	+	+	+
NAS	-	-	-	-
LAB	-	-	-	+
COR	-	-	-	-
DORS	-	-	-	+
SGLOT	-	-	-	-
ANT	-	-	-	-
SON	+	+	+	+
VCE	+	+	+	+
CONT	+	+	+	+
HIGH	-	+	-	+
LOW	+	-	-	-
BACK	+	-	-	+
ROUND	-	-	-	+
LAT	-	-	-	-
STRID	-	-	-	-

3.3.3.1 *Diphthong Inventory Constraints*

The following undominated constraints help to determine the vowel inventory of Malagasy:

(13) *Contour([e], [i]):

There is no [e͡i] diphthong. Short form: *|ei|

(14) *Contour([+high], [-high]):

All diphthongs are rising. Short form: *|[+high][-high]|.

(15) *Contour([-low], [+low]):

Once again, all diphthongs are rising. Short form: *|[-low][+low]|.

3.3.4 general segment inventory constraints

The following undominated constraints keep vowels and consonants from mixing within a segment.

(16) *Contour(C, V), *Contour(V, C):

Any given segment is either a consonant or a vowel, but not a complex segment that encompasses both. Short forms: *|CV|, *|VC|.

3.4 THE BASIC STRESS PATTERN

The general stress pattern for Malagasy is right-to-left trochaic with the rightmost stress of a word having special prominence, as can be seen in (17). Note the forms in the bottom row of the table, which constitute evidence, albeit imperfect (they are not monomorphemic), against an analysis in which the primary stress is assigned to the penultimate syllable and secondary stresses are assigned in trochees from left to right.

(17)

Gloss	Form	Gloss	Form
to go	manˈdeha	*to see*	maˈhita
to wait	miˈandr͡i	*to work*	miˈasa
to sun-dry	miˈhahi	*cotton*	ˌlandiˈhazu
to drink	miˈsutʃu	*the brain*	ˌatiˈduha
to visit	maˈmaŋgi	*bamboo*	ˌbaraˈrata
work together	miˌfampiˌrahaˈraha	*love each other*	miˌfaŋkaˈtia

3.4.1 RIGHT-TO-LEFT TROCHAIC STRESS

The right-to-left trochaic pattern exemplified in (17) is analyzed here by a *CLASH/*LAPSE account similar to that of Gordon (2002) (see Appendix D for a comparison of Gordon's approach with the approach taken here). This type of stress analysis has a number of advantages over a foot-based account. Explicit metrical feet would bring an unwarranted complexity into the representation and require constraints of alignment; the Generalized Alignment constraints generally used for this purpose are demonstrably overly powerful

(Eisner 1997d). For a defense of this general approach from a typological standpoint see Gordon (2002) and works cited therein. The constraints needed for a *Clash/*Lapse account are as follows:

(18) *Sequence([V,+stress], [V,+stress], C):

Short form — *Clash. *Output a violation for each sequence of adjacent stressed vocalic sonority peaks.* That is, a violation is incurred by each instance of a stressed vowel segment followed by another stressed vowel segment, where zero or more consonants may intervene. Note that a vowel segment is made up of one or more vowel symbols bounded on either side by segment boundary symbols; a diphthong, for example, is treated as a single vowel segment or a single vocalic sonority peak in this representation, although it is made up of more than one vocalic symbol.

(19) *Sequence([V,-stress], [V,-stress], C):

Short form — *Lapse. *Output a violation for each sequence of adjacent unstressed vocalic sonority peaks.*

(20) *Final([V,+stress], C):

Short form — NonFinality. *Output a violation if the final vowel segment (recall that a diphthong is a single vowel segment in this representation) of a word carries stress.*

The operation of each of these constraints in a typical penultimate-stressed output is shown in (21).

(21) /landihazu/→[ˌlandiˈhazu]$_s$ 'cotton'

/landihazu/	NonFinality	*Clash	*Lapse
☞ ˌlandiˈhazu			
landiˈhazu			*!
lanˌdiˈhazu		*!	
lanˌdihaˈzu	*!		

3.4.2 PRIMARY STRESS ASSIGNMENT

The final stress of a word carries special prominence in Malagasy, and is therefore referred to as primary, with other stresses being referred to as secondary. This distinction is present at all levels of representation, and is analyzed in the same way in all of them. The cause of this stress pattern is a trio of constraints which together might be called "Culminativity with Rightward Alignment." The first of the three constraints, given in (22), has the effect of forcing the rightmost stress in the word to be primary, *i.e.* it gives the final syllable special prominence. A language with leftmost primary stress would employ the constraint given in (23). The second constraint (24) discourages multiplication of primary stresses. When outranked by *RtMost2ndary, this makes sure that all stresses except the rightmost are secondary. Finally, constraint (25) ensures that every word has at least one stressed vowel (note that this is necessary only for one-syllable words, since existence of stress is otherwise ensured by *Lapse).

(22) *Final([+stress,-prim], [-stress]) :

Short form—*RTMOST2NDARY. *Output a violation for any word in which the rightmost stress is secondary.*

(23) *INITIAL([+STRESS,-PRIM], [-stress]) :

Short form — *LFTMOST2NDARY. *Output a violation for any word in which the initial stress is secondary.*

(24) *([+PRIM]) :

Output a violation for each segment with primary stress.

(25) EXISTS([+STRESS]) :

Output a violation for any word with no segments marked [+STRESS].

I will use the same example as for the basic pattern of stress location to show how the primary stress assignment constraints operate, in (26).

(26) /landihazu/→[ˌlandiˈhazu]$_s$ 'cotton'

/landihazu/	*RTMOST2NDARY	*[+PRIM]
☞ ˌlandiˈhazu		*
ˈlandiˈhazu		**!
ˌlandiˌhazu	*!	
ˈlandiˌhazu	*!	*

3.4.3 EXCEPTIONS

Exceptions to the basic stress pattern are discussed in §§3.5–3.8.

3.5 THE LEXICAL/POST-LEXICAL DISTINCTION

3.5.1 WEAK AND PSEUDO-WEAK ROOTS: DATA SUMMARY

3.5.1.1 *Pre-Vocalic Consonant Neutralization*

Malagasy has a class of roots that Keenan & Polinsky (1998) refer to as the *weak* and *pseudo-weak* roots. These roots have a number of interesting characteristics. First, as shown in Table 3.1 on page 129, and summarized in (27),

(27)

Present	Circumstantial
[n]	[n]
[n]	[m]
[ʧ]	[f]
[ʧ]	[t]
[ʧ]	[ʧ]
[ʧ]	[r]
[k]	[f]
[k]	[k]
[k]	[h]

the present indicative active form of these roots (this is an unsuffixed form) has only three endings: [-n] ((a)–(c), (g), (h)), [-ka] ((d), (e), (m)–(o)), and [-ʧa] (the rest). The circumstantial form, however, has eight[14]. The simplest explanation here is that the nine different present-circumstantial pairings involving weak roots have nine different underlying forms, and that the distinctions inherent in these underlying forms are neutralized to form the present tense. This is an extremely unusual finding cross-linguistically—consonant neutralization in a prevocalic environment.

[14] [-nan], [-man], [-fan], [-tan], [-ran], and [-han] are shown in Table 3.1, which is from Erwin (1995). Erwin (1995) asserts the existence of the other hypothesized endings ([-kan] and [-ʧan]), but thus far I have not been able to find any examples. Note that the forms in the table are given without secondary stress, as in Erwin's (1995) original.

	Present (*m*+stem)	Circumstantial (stem+*an*)	Gloss
(a)	maˈnan͡dran	ananˈ͡draman	*to try*
(b)	ˈmin͡dran	inˈ͡draman	*to borrow*
(c)	miˈtan͡drin	itanˈ͡dreman	*to care for*
(d)	maŋˈgataka	aŋgaˈtahan	*to ask for*
(e)	maˈnaraka	anaˈrahan	*to follow*
(f)	miˈanatʃa	iaˈnaran	*to study*
(g)	miˈhinan	ihiˈnanan	*to eat*
(h)	miˈtsaŋgan	itsaŋˈganan	*to stand*
(i)	miˈsa͡ut͡ʃa	iˈsa͡uran	*to thank*
(j)	maˈnan͡dratʃa	ananˈ͡dratan	*to promote*
(k)	mahafiˈnaritʃa	ahafinaˈretan	*to please*
(l)	manˈ͡drakutʃa	an͡draˈkufan	*to cover*
(m)	maˈnahaka	anaˈhafan	*to scatter*
(n)	maˈnaluka	anaˈlufan	*to shade*
(o)	miˈlelaka	ileˈlafan	*to lick*

Table 3.1: Pre-Vocalic Consonant Simplification

3.5.1.2 *Aberrant Stress*

Looking at the table it is possible to notice another interesting fact about the present tense forms—many of them ((d)–(f), (i)–(o)) have antepenultimate stress, contrary to the basic stress pattern of the language (§3.4). The nasal-final elements of the table ((a)–(c), (g), (h)) appear normal, having penultimate stress, but in fact nasal-final roots other than weaks and pseudo-weaks generally carry final stress:

(28)

Gloss	Form
soften, weaken	ˌmanaˈlem
pleasant conversation	kuˈran
tree with reddish wood	laˈlun
forget	haˈdin

3.5.1.3 *Compounds*

The final clue comes from genitives and reduplicated forms.

Normal Vowel-Initial Compounds Normally a genitival construction where the possessor begins with a vowel consists of the object of possession, followed by [n], and finally the possessor:

(29)

Gloss	Object	Possessor	Genitive
someone's clothes	aˈkandzu	ˈulun	aˌkandzuˌnulun
Vao's money	ˈvula	iˈvâu	ˌvulaniˈvâu

Reduplicated forms are similar, dispensing only with the linking [n]:

(30)

Gloss	Isolation Form	Reduplicated
understood	ˈazu	ˌazuˈazu
high	ˈavu	ˌavuˈavu

Normal Consonant-Initial Compounds In a normal genitive (*i.e.,* one in which the object of possession is not a weak or pseudo-weak root) where the possessor begins with a consonant, the construction consists of the object, followed by a nasal whose place of articulation is the same as the first consonant of the possessor, followed by the possessor, whose first consonant has been changed to the closest affricate, or a stop if there is no affricate at the same point of articulation. This is illustrated by the following examples (see §3.7.1.3 for more information on this sort of data):

(31)

Gloss	Object	Possessor	Genitive
Rabe's father	ˈr͡aɪ	raˈbe	ˌr͡aɪnd͡raˈbe
plum (foreigner's peach)	ˈp͡aɪsu	vaˈzaha	ˌp͡aɪsumbaˈzaha
Rabe's money	ˈvula	raˈbe	ˌvulan͡draˈbe

Normal reduplication (where the reduplicant is not a weak or pseudo-weak root) of short consonant-initial roots involves a simple concatenation of the root with its copy:

(32)

Gloss	Simple	Reduplicated
big	ˈbe	ˌbeˈbe
tell lies	ˈlaĩŋga	ˌlaĩŋgaˈlaĩŋga
visit	ˈvaŋgi	ˌvaŋgiˈvaŋgi
continue	ˈtuhi	ˌtuhiˈtuhi

Compounds with Nasal-Final Weak Roots A genitival construction involving a nasal-final weak or pseudo-weak root is indistinguishable from a normal genitive except for the stress pattern of the object:

(33)

Gloss	Object	Possessor	Genitive
Rabe's month	ˈvulan	raˈbe	ˌvulan͡draˈbe

In reduplication, however, there is a distinction between nasal-final weak and non-weak roots. The weak roots reduplicate to an output that looks like a genitive, with the nasal hardening found in (31). This is seen in (34).

(34)

Gloss	Simple	Reduplicated
baggage	ˈentan	ˌentaˈnentan
some	ˈsasan	ˌsasanˈtsasan
alive	ˈvelun	ˌvelumˈbelun

Reduplication of nasal-final non-weak roots is quite different—a linking vowel emerges between the base and the reduplicant, a vowel that is unpredictable from the isolation form, except in extremely careful speech, where it sometimes appears word-finally. Ex-

amples are given in (35). The details of Malagasy partial reduplication ([haˌdinuˈdinu] as opposed to *[haˌdinuhaˈdinu]) will be covered in §3.11.

(35)

Gloss	Simple	Reduplicated
forget	haˈdin	haˌdinuˈdin
wander about	ˈren	ˌreniˈren

Compounds with Oral-Final Weak Roots In genitives where the object of possession is a weak root whose final consonant is oral, the linking nasal does not appear. If the possessor begins with a consonant, the final consonant of the object of possession disappears and the initial consonant of the possessor is hardened to the closest affricate or stop (full analysis of this phenomenon will be given in §3.7.3.4), as in (36), where part (a) shows vowel-initial possessors and part (b) shows consonant-initial.

(36)

	Gloss	Object	Possessor	Genitive
(a)	*chicken's foot*	ˈtuŋgutʃa	aˈkuhu	ˌtuŋgutʃaˈkuhu
	Soa's shoulder	ˈsuruka	iˈsua	ˌsurukiˈsua
(b)	*a child's shoulder*	ˈsuruka	ˈzaza	ˌsuruˈʣaza
	foot of a bed	ˈtuŋgutʃa	ˌfaraˈfara	ˌtuŋguˌparaˈfara

Reduplication of weak oral-final roots comes out the same way:

(37)

Gloss	Simple	Reduplicated
shade	ˈaluka	ˌaluˈkaluka
bouncing back	ˈevutʃa	ˌevuˈʧevutʃa
writing	ˈsuratʃa	ˌsuraˈtsuratʃa
conversation	ˈresaka	ˌresaˈd͡resaka
selling	ˈvarutʃa	ˌvaruˈbarutʃa
thing	ˈzavatʃa	ˌzavaˈʣavatʃa

3.5.1.4 *Pre-Theoretical Analysis*

To summarize the data given above, weak roots appearing in isolation carry primary stress one syllable earlier than non-weak roots (pseudo-weak roots are those that are too short for this to be the case, but otherwise act like weak roots). They behave in compounds as though they are consonant-final, and, finally, they exhibit a reduced consonant inventory in isolation. A mono-stratal analysis for these facts would of necessity be rather complicated, but multi-stratal theories such as LPM-OT or classical rule ordering offer a simpler explanation: at an earlier level (the lexical level), weak and pseudo-weak roots are consonant-final. At this earlier level the final consonants are not protected by a final vowel and thus become simplified. Stress is assigned at the earlier level as well, following the basic stress pattern described in §3.4, with some significant exceptions to be discussed later (§3.8). At the top (post-lexical) level, weak and pseudo-weak oral-consonant-final roots are supplied with an epenthetic final vowel which does not affect the stress pattern, so the primary stress on weak roots is antepenultimate (penultimate for pseudo-weak roots,

which are underlyingly one syllable long). A post-lexical process removes vowels after a word-final nasal, so non-weak roots lose the vowel that protected their final nasal consonant from being simplified at the lexical level and thus their primary stress appears one syllable to the right of their weak-root counterparts. This vowel appears in combined forms for the non-weak roots, as in the reduplication examples of (35).

Now that the justification for post-lexical and lexical levels has been established, it remains to see how the phenomena under discussion may be accounted for in this LPM-OT analysis. Because the purpose of this section is simply to establish the overall architecture of the system, only the simplest of these phenomena (the post-nasal apocope and post-oral epenthesis) will be analyzed here. The rest will be dealt with in §3.7.

3.5.2 ILLUSTRATION OF THE ARCHITECTURE

Before delving into the details of the Optimality Theoretic analysis, it may be useful to look at some derivations and see how the system is claimed to work. Below are given derivations of ordinary stems whose final consonant is a nasal, followed by weak nasal stems, ordinary oral-final stems, and finally weak oral-final stems.

3.5.2.1 *Non-Weak Nasal Stems*

An ordinary stem whose final consonant is a nasal has a derivation similar to that given in (38) for *saˈlam* 'healthy.'

(38)

Underlying	/salama/
Lexical	[saˈlama]$_w$
Post-Lexical	[saˈlam]

Essentially, the underlying form and the lexical form have a final vowel, and this vowel is removed in the post-lexical form (notice the lack of stem-final neutralization). A reduplication of another ordinary stem whose final consonant is a nasal appears in (39).

(39)

	Simple	Reduplicated
Underlying	/hadinu/	/hadinu+RED/
Lexical	[haˈdinu]$_w$	[haˈdinu]$_w$+[ˈdinu]$_w$
Post-Lexical	[haˈdin]	[haˌdinuˈdin]

Here the simple derivation for *haˈdin* 'to forget' follows the same pattern as the derivation for *laˈlun* above, but reduplication allows the final vowel of the stem to survive. Reduplication will be investigated in more detail in §3.11.

3.5.2.2 *Weak Nasal Stems*

A nasal-final weak stem such as *and͡ram* 'to try' comes out looking somewhat similar to an ordinary nasal stem because the effect of post-nasal apocope erases some of the distinction between them, but such stems have final neutralization and other effects given above. Here, in (40), is a derivation of the present indicative active form, which shows final neutralization, and the circumstantial form, which does not.

(40)

	Present Active	Circumstantial
Underlying	/m+an+and͡ram/	/an+and͡ram+an/
Lexical	[man'and͡ran]$_w$	[anan'd͡raman]$_w$
Post-Lexical	[man'and͡ran]	[anan'd͡raman]

Note that the stress lapse found in the circumstantial form here is analyzed further in §3.6.

3.5.2.3 *Non-Weak Oral Stems*

Stems that end underlyingly in an oral consonant followed by a vowel are not subject to much phonological change of the sort described above. For example, *ma'hita* 'see' is unchanged from its underlying form:

(41)

Underlying	/m+a+hita/
Lexical	[ma'hita]$_w$
Post-Lexical	[ma'hita]

Compounding behaves similarly here as well (*ˌlâiŋga'lâiŋga* 'tell lies'):

(42)

	Simple	Reduplicated
Underlying	/lâiŋga/	/lâiŋga+RED/
Lexical	['lâiŋga]$_w$	['lâiŋga]$_w$+['lâiŋga]$_w$
Post-Lexical	['lâiŋga]	[ˌlâiŋga'lâiŋga]

3.5.2.4 *Weak Oral Stems*

The weak stems are, as stated before, underlyingly consonant final. At the postlexical level a final vowel is added, but at the lexical level, before the epenthesis, the final consonant undergoes neutralization. In compounds the final consonant of the first part of the compound interacts with the initial consonant of the second part. All of these phenomena can be seen in the paradigm of *ˈfantaʧa* 'known,' seen in (43).

(43)

	Present	Passivized	Reduplicated	Passivized Redup.
UR	/fantaɾ/	/fantaɾ+in/	/fantaɾ+RED/	/fantaɾ+RED+in/
Lex.	[ˈfantaʧ]$_w$	[fanˈtaɾin]$_w$	[ˈfantaʧ]$_w$+[ˈfantaʧ]$_w$	[ˈfantaʧ]$_w$+[fanˈtaɾin]$_w$
P-Lex.	[ˈfantaʧa]	[fanˈtaɾin]	[ˌfantaˈpantaʧa]	[ˌfantapanˈtaɾin]

The analysis underlying these derivations will be worked out in more detail in §3.7 (for the consonant patterns) and §3.11 (reduplication), but essentially in the present tense form the final consonant is in word-final position at the lexical level and becomes neutralized, and a final [a] is inserted at the post-lexical level. With the passive suffix appended, the stem-final consonant is protected and remains unchanged. In the reduplicated forms, the stem-final consonant in the base has changed from a continuant, as in the underlying form, to an affricate at the lexical level. The [-CONT] feature of the affricate merges with the labial place of the stem-initial consonant to form a labial stop.

3.5.3 POST-NASAL APOCOPE

As stated in §3.5.1.4, my analysis for the data pattern of roots whose final consonant is a nasal is that each form's stress pattern is established at the lexical level and subsequently

final unstressed vowels are removed at the post-lexical level[15]. This proposal is illustrated in (44), which shows the proposed output of the lexical level and how it is transformed to the actual surface form.

(44)

	Gloss	Hyp. Lex. Output	Surface Form
(a)	*give*	[ˌmanuˈme]$_w$	[ˌmanuˈme]
	do	[maˈna͡u]$_w$	[maˈna͡u]
(b)	*believes*	[ˈminu]$_w$	[ˈmin]
	soften, weaken	[ˌmanaˈleme]$_w$	[ˌmanaˈlem]
	tree with reddish wood	[laˈluna]$_w$	[laˈlun]
	forget	[haˈdinu]$_w$	[haˈdin]
(c)	*baggage*	[ˈentan]$_w$	[ˈentan]
	to borrow	[ˈmind͡ran]$_w$	[ˈmind͡ran]
	eat	[miˈhinan]$_w$	[miˈhinan]
	take care of	[miˈtand͡rin]$_w$	[miˈtand͡rin]

Final post-nasal vowels are deleted due to the effect of the following constraint:

(45) *FinalSequence([+nas],V,∅):

Output a violation for any word ending on the surface in a sequence consisting of a nasal immediately followed by a vowel. Short form—*NV#.

*NV# is violated in the output in some cases, due to preservation of final vowels that carry stress at the lexical level. This preservation is driven by MaxIO([+stress]):

[15]How final vowels come to be stressed in some words will be explored in §3.8.

(46) MaxIO([+stress]):

Output a violation for each underlyingly stressed vowel (here, this means stressed at the lexical level) that does not correspond to a segment on the surface level.

Further, it cannot be satisfied by inserting an extra segment at the end of the word (*i.e.*, by violating DepIO) or by deleting the nasal (MaxIO(C)).

(47) DepIO:

Output a violation for any segment in the surface level that has no underlying correspondent. An abbreviation for the two constraints DepIO(V) and DepIO(C).

(48) MaxIO(C):

Output a violation for any underlying consonant that does not correspond to a surface segment.

Finally, many of the forms above satisfy *NV# at the expense of allowing vowel deletion (thus violating MaxIO(V) (49)) and word-final stress (NonFinality).

(49) MaxIO(V):

Output a violation for any underlying vowel that does not correspond to a surface segment.

The postlexical constraint ranking that accounts for the phenomenon is (50)[16]

(50) DepIO, MaxIO([+stress]), MaxIO(C) $\gg_p$ *NV# $\gg_p$ MaxIO(V), NonFinality

[16] Recall that "$\gg_p$" means "outranks postlexically."

as can be seen in tableaux (51)–(53).

(51) /a͡ina/ → [ˈa͡ina]$_s$ → [ˈa͡ina]$_w$ → [ˈa͡in] ‘breath’[17]

[ˈa͡ina]$_w$	MaxIO(C)	*N V#	NonFinality	MaxIO(V)
☞ ˈa͡inØ			*	*
ˈa͡ina		*!		
ˈa͡iØa	*!			

(52) /m+an$_x$+uˈmez/ → /m/+[ˌan$_x$uˈmez]$_s$ → [ˌman$_x$uˈme]$_w$ → [ˌmanuˈme] ‘*pres.*+*act.*+give’[18]

[ˌman$_x$uˈme]$_w$	MaxIO([+stress])	*N V#	NonFinality	MaxIO(V)	*[-low]
☞ ˌmanuˈme		*	*		**
ˈmanumØ	*!			*	*

(53) /laluna/ → [laˈluna]$_s$ → [laˈluna]$_w$ → [laˈlun] ‘a tree with hard reddish wood’

[laˈluna]$_w$	DepIO(V)	*N V#	NonFinality	MaxIO(V)	*Lapse
☞ laˈlunØ			*	*	
laˈluna		*!			
laˈlunaa	*!				*

[17]Here and in the tableaux to follow I have used the symbol ‘Ø’ to denote a deleted segment.

[18]The symbol n$_x$ represents an abtract underlying segment belonging only to this morpheme. It will be explained in §3.7.2.2.

See Blevins (1997) for similar data in Gilbertese.

Note that DepIO(C) and MaxIO([+stress]) are undominated at the postlexical level.

3.5.4 post-oral final vowel epenthesis

To recap, my analysis is that weaks and pseudo-weaks are underlyingly consonant-final, and they continue to be so at the output of the lexical level. Compounding takes place post-lexically, and its input appears to be consonant-final when weak roots are involved. In all cases, not just compounding, the stress pattern established by the lexical level is preserved at the post-lexical level, but oral consonants are not allowed word-finally, so the vowel [a] is epenthesized after the final consonant. It does not disturb the stress pattern, so antepenultimate stress results if the output of the word level had penultimate stress.

In the epenthetic forms the final consonant is not deleted but instead epenthesis occurs, because the undominated ban on deleting oral consonants (54) outranks the ban on inserting vowels and (subsequently) creating a stress lapse, as shown in ranking (55).

(54) MaxIO([-nas,C]):

Output a violation for each underlying oral consonant that does not correspond to a surface segment.

(55) $\text{MaxIO}([\text{-nas,C}]) \gg_p \text{DepIO(V)}, \text{*Lapse}$

Stress is not assigned to the inserted vowel because it is word-final.

(56) $\text{NonFinality} \gg_p \text{*Lapse}$

Finally, the vowel inserted is always [a]. I analyze this as being the result of a constraint banning non-low vowels (57). This constraint has no effect on vowels with an underlying basis due to *[-LOW] being outranked by a bank of constraints enforcing faithfulness to vowel place features (58).

(57) *[-LOW]:

Output a violation for each surface non-low vowel.

(58) IDENTIO(V-PLACE):

Output a violation for any underlying segment whose surface correspondent differs from it in one of more place features. Note that this is actually an abbreviation for a sequence of constraints whose ranking with respect to one another is irrelevant to the phenomenon at hand: IDENTIO([±LOW]), IDENTIO([±HIGH]), IDENTIO([±BACK]), and IDENTIO([±ROUND]).

And of course MAXIO(V) must be active to prevent all vowels other than [a] from being deleted. The ranking that establishes all of this behavior is shown in (59).

(59) MAXIO(V), IDENTIO(V-PLACE) $\gg_p$ *[-LOW]

Rankings (55), (56), and (59) are all demonstrated in tableau (60).

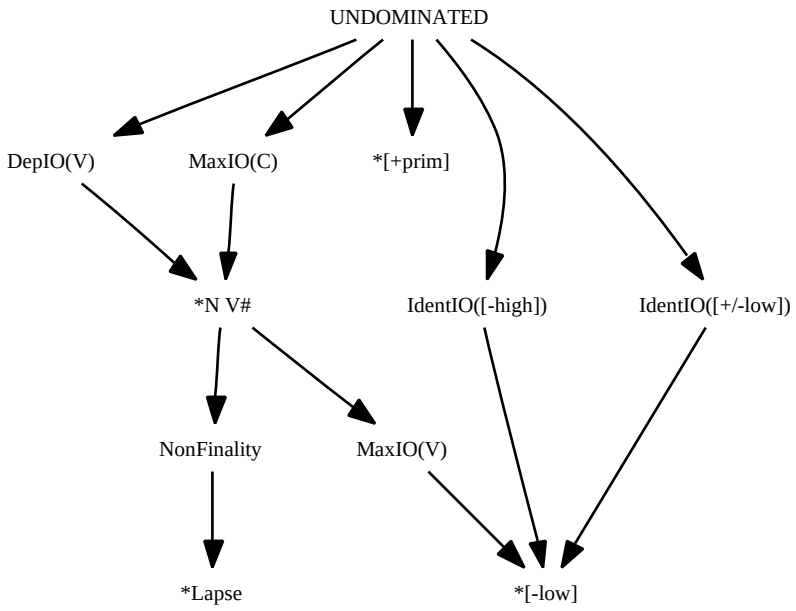
(60) /aluk/ → [ˈaluk]$_s$ → [ˈaluk]$_w$ → [ˈaluka] 'shade'

	$[\text{ˈaluk}]_w$	MaxIO([-nas, C])	IdentIO([±V-place])	*[-nas, C]#	MaxIO(C)	DepIO(V)	NonFinality	*[-low]	*Lapse
☞	ˈaluka					*		*	*
	ˈaluku					*		**!	*
	ˈaluki					*		**!	*
	ˌaluˈka					*	*!	*	
	ˈaluk			*!				*	
	ˈalaka		*!			*			*
	ˈaluØ	*!			*			*	

3.5.5 RANKINGS SO FAR

So far it has been stated that the following constraints are undominated at all levels: *|[+son] [−son]|, *|[αpl,C] [−αpl,C]|, *|[−cont] [+cont, −cor]|, *|[αnas] [−αnas]|, *|[+cont] [−cont]|, *|[αvce] [−αvce]|, *t̲ X, *t̲#, *X ʃ, *#ʃ, *|ei|, *|[+high] [−high]|, *|[−low] [+low]|, *|CV|, and *|VC|. In addition all levels have the ranking *Rtmost-2ndary ≫ *[+prim].

The current section added to this some rankings for the post-lexical level. Here the constraints DepIO(C), MaxIO([+stress]), and MaxIO([−nas,C]) were asserted to be undominated, and other rankings were given consistent with the following Hasse diagram:



In order to try to keep the diagrams readable, I have omitted mention of undominated constraints in the ranking diagrams given here and elsewhere in this text.

3.6 THE WORD/STEM LEVEL DISTINCTION

Section §3.5 went into the justification for having a grammar for lexical phonology and a different grammar for post-lexical phonology. The analysis given here is not composed of two grammars (lexical and post-lexical), however, but three. There are two classes of affixes in Malagasy — one that binds tightly to its stem and has idiosyncratic allomorphs, and one that binds less tightly and is more uniform across words (see §3.2.4 for some examples). The behavior of these two classes of affixes suggests, as will be seen below, a division of the lexical grammar into a stem grammar, which comes first, and a word grammar, which follows it.

3.6.1 WORD-LEVEL SUFFIXES

The rightmost stress of a word—and only that stress—is shifted to the right after the addition of any one of a class of suffixes including *-a,* the imperative suffix, and *-an*, the circumstantial suffix (referred to in some places, *e.g.* Erwin (1995), as the relativizing suffix). This behavior is shown in the following table[19].

(61)

	Present Active	Circumstantial	Proposed UR of Circ.
(a) *sleeps*	maˈturi	matuˈria	/m+a+turi+a/
(b) *sits*	miˈpetʃaka	mipeˈtʃaha	/m+i+petʃah+a/
(c) *sings*	miˈhira	mihiˈra	/m+i+hira+a/
(d) *arranges*	ˌmanamˈbuatʃa	ˌmanambuˈara	/m+an$_x$+ambuar+a/
(e) *goes home*	ˈmudi	muˈdia	/m+udi+a/
(f) *works*	miˈasa	miaˈsa	/m+i+asa+a/

The suffixes that take part in this data pattern appear at the extreme right edge of a word, and have the property of being regular in the sense of having a single allomorph regardless of the stem to which they are attached. Because stress assignment is driven by constraints on allowable stress patterns at the right edge of a word, no prefixes induce stress shift, but there are a number of prefixes that occur at the extreme left edge of words, with no allomorphs, and with, in some cases, lexicalized stress patterns. I would class these prefixes as members of the same set of affixes, which I will call, for reasons to be made clear later, *word-level affixes*.

[19] Note that forms (c) and (f) exhibit vowel merger with opaque stress, which will be analyzed in §3.8.1.

3.6.2 STEM-LEVEL SUFFIXES

The affixation of other suffixes leads to a different pattern of stress shift.

(62)

	Active	Passive	Proposed UR of Pass.
(a) *to disorder*	saˈritaka	ˌsariˈtahin	/saritah+in/
(b) *to overthrow*	kuˈruntan	ˌkurunˈtanin	/kuruntan+in/

These affixes appear, essentially, to create a new stem that continues to obey the basic stress pattern of the language. Stem-level affixes, as I will call this class, tend to be more idiosyncratic than the word-level affixes, with multiple stem-specific allomorphs (for example, the passive suffix above appears as *-an* or *-in* depending on the stem), and although suffixes appear not to stack in Malagasy, the stem-level prefixes appear to the interior of the word-level prefixes (the stem-level prefixes include, for example, the active marker *an-/ana-/i-/a-/∅-*.) Stem-level prefixes, unlike word-level prefixes, never carry a phonemic stress.

3.6.3 THE THREE-LEVEL ARCHITECTURE

My analysis for these facts is that there is a *stem* level of grammar in which the basic stress pattern described in §3.4 is followed without much exception (see §3.8 for the inevitable complications). The output of the stem level grammar is fed into the *word* level. At this level the basic stress pattern constraints are still operative, but they are partially superceded by faithfulness to the stress pattern established at the stem level. §3.5 referred to a "lexical output." The word and stem levels together make up the lexical level, so the output of the

lexical level is the same as the output of the word level.

The behavior of the stem-level forms is covered by §§3.4–3.5 (basic stress plus final epenthesis for [saˈɾitaka]), so all that needs elaboration is the word-level stress shift.

3.6.3.1 *Word-Level Stress Shift*

Stress shift caused by word-level suffixes (these include the imperative ending -a and the circumstantial ending -an, as opposed to the passive ending -in/-an) can create lapse. This is because a stress must appear on the newly penultimate syllable (due to undominated *FinalLapse and high-ranked NonFinality). If the stress on what had been the penultimate syllable were to be preserved, a stress clash would result. Since a stress clash does not appear in the actual forms, the desire to avoid a clash must outrank the desire to preserve stress:

(63) IdentIO([+stress]):

Output a violation for each underlyingly stressed segment that has a unstressed surface correspondent.

hence ranking (64), demonstrated in tableau (65).

(64) NonFinality, *Clash $\gg_w$ IdentIO([+stress])

(65) /m+udi+a/→/m/+['udi]$_s$+/a/→[mu'dia]$_w$ *'pres.*+go home+*imp.'*

/m/+['udi]$_s$+/a/	*FinalLapse	NonFinality	IdentIO([-stress, V])	*Clash	*Lapse	IdentIO([+stress])
☞ mu'dia			*			*
ˌmu'dia			*	*!		
ˌmudi'a		*!	*			
'mudia	*!				*	

Additionally, the syllable that had been antepenultimate continues not to receive a stress, thus creating a lapse. The desire to preserve lack of stress in a syllable (66) outranks the desire to avoid lapse, as seen in ranking (67) and tableau (68).

(66) IdentIO([V,-stress]):

Output a violation for each underlyingly unstressed vowel segment that has a stressed surface correspondent.

(67) IdentIO([V,-stress]) $\gg_w$ *Lapse

(68) /m+a+turi+a/→/m/+[a'turi]$_s$+/a/→[matu'ria]$_w$ *'pres.*+*act.*+sleep+*imp.'*

/m/+[a'turi]$_s$+/a/	IdentIO([V,-stress])	*Lapse
☞ matu'ria	*	*
ˌmatu'ria	**!	

Stress preservation is outweighed by vowel preservation, however:

(69) MaxIO(V) ≫$_w$ IdentIO([V,-stress]), IdentIO([+stress])

(70) /an$_x$+dihiz+an/ → /an$_x$/+[ˈdihiz]+/an/ → [an$_x$diˈhizan]$_w$ '*act.*+dance+*circum.*'[20]

/an$_x$/+[ˈdihiz]$_s$+/an/	MaxIO(V)	IdentIO([-stress,V])	*Lapse	IdentIO([+stress])
☞ an$_x$diˈhizan		*	*	*
an$_x$ˈdihØzan	*!			

Notice that both *Lapse and IdentIO([+stress]) could be avoided in longer forms by creating a clash, but this does not occur, as shown in (72). It appears, then, that the following ranking exists:

(71) *Clash ≫$_w$ *Lapse

(72) /aha+fali+an/ → /aha/+[ˈfali]$_s$+/an/ → [ˌahafaˈlian]$_w$ '*act.*+make happy+*circum.*'

/aha/+[fali]+/an/	*Clash	*Lapse	IdentIO([+stress])
☞ ˌahafaˈlian		*	*
ˌahaˌfaˈlian	*!		

[20]The annotation "/n$_x$/" here indicates the lexical marking feature [+x], discussed in §3.7.2.2.

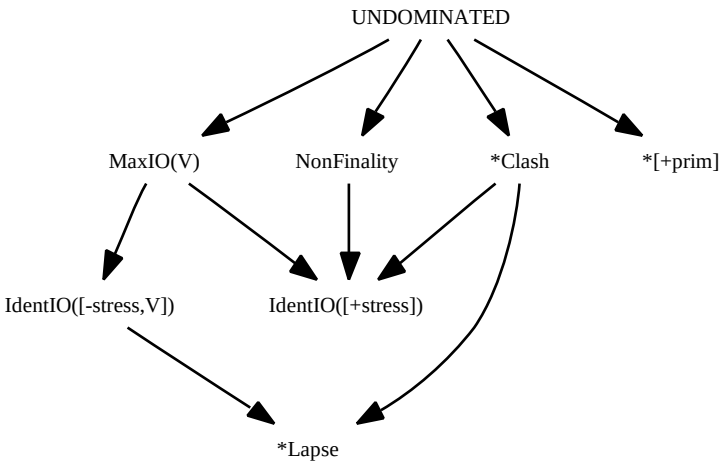
3.6.4 LOOKING AHEAD

The analysis of Malagasy will henceforth be the discovery of three different constraint rankings: stem level, word level, and post-lexical. The overall picture is that stem-level affixes are added at the stem level, word-level affixes are added at the word level, and compounds are put together at the post-lexical level.

In the above sections I have attempted to justify a stem-word-postlexical approach to Malagasy analysis. In so doing I have introduced a number of complex phenomena without fully analyzing these phenomena. Below I will first explore the weak and pseudo-weak stems in more detail, then move on to the fuller details of the Malagasy stress system. Later, I will introduce a new set of consonant-final stems where the final consonants undergo deletion rather than simplification. Next will be vowels that weaken. Wrapping up the analysis of Malagasy will be reduplication and compounding.

3.6.5 RANKINGS SO FAR

At this point, undominated constraints and rankings for all levels other than word are unchanged from those given in §3.5.5. The word level now has the following known rankings:



3.7 ANALYSIS OF WEAK AND PSEUDO-WEAK STEM BEHAVIORS

Now that the stem-word-postlexical architecture has been established, it remains to see how the phenomena under discussion may be accounted for in this LPM-OT analysis. I will begin with the nasal-final weak and pseudo-weak roots and their behavior in isolation and compounds and proceed from there to the oral-final roots.

3.7.1 NASAL-FINAL WEAK AND PSEUDO-WEAK ROOTS

There are two types of nasal-final weak roots: in the first (forms (a)–(c) below) the nasal appears as [m] in the suffixed form, and in the other (forms (d) and(e)) it appears as [n].

(73)

	Present (*m*+stem)	Circumstantial (stem+*an*)	Gloss
(a)	m-aˈn-and͡ɾan	an-anˈd͡ɾam-an	*to try*
(b)	ˈm-ind͡ɾan	inˈd͡ɾam-an	*to borrow*
(c)	m-i-ˈtand͡ɾin	i-tanˈd͡ɾem-an	*to care for*
(d)	m-i-ˈhinan	i-hiˈnan-an	*to eat*
(e)	m-i-ˈtsaŋgan	i-tsaŋˈgan-an	*to stand*

The obvious choice, then, is to analyze the underlying form of the final nasal in the first set as /m/ and the second as /n/.

3.7.1.1 *Word-Final Simplification*

I posit that underlying /m/ becomes [n] when word-final because of a cross-linguistic dispreference for labial consonants (74), which in Malagasy appears to dominate general preservation of labial place (75) as shown in ranking (76).

(74) *([LAB, C]):

Output one violation for each instance of a surface labial consonant.

(75) IDENTIO([LAB]):

Output a violation for each underlyingly [+LAB] *segment that has a surface correspondent which is* [-LAB].

(76) $*[\text{LAB,C}] \gg_w \text{IDENTIO}([\text{LAB}])$

However, an underlying labial release into a vowel is preserved via the undominated constraint IDENTIO(LAB)/_V (77) as well as MAXIO(C)/_V (78), so underlying pre-vocalic /m/ remains as such thanks to the ranking given in (79).

(77) IDENTIOBEFORE([LAB], V):

Short form—IDENTIO([LAB])/_V. *Output a violation if an underlyingly labial segment, followed by an underlying vowel, changes to non-labial.*

(78) MAXIOBEFORE(C, V):

Short form—MAXIO(C)/_V. *Output a violation for each instance of an underlyingly* [-SYL] *segment that immediately precedes, in the underlying form, a* [+SYL] *segment if the underlying* [-SYL] *segment does not have a surface correspondent.*

(79) MAXIO(C)/_V, IDENTIO([LAB])/_V $\gg_w$ *[LAB, C].

The rankings above are illustrated in tableaux (80) and (81).

(80) /m+i+tandrem/ → /m/+[i'tandrem]$_s$ → [mi'tandrin]$_w$ '*pres.*+*act.*+take care of'

/m/+[i'tandrem]$_s$	*[LAB, C]	IDENTIO([LAB])
☞ mi'tandrin	*	*
mi'tandrim	**!	

(81) /i+tandʒ̆rem+an/ → [i'tandʒ̆rem]$_s$+/an/ → [itan'dʒ̆reman]$_w$ '*act.*+take care of+*circum.*'

[i'tandʒ̆rem]$_s$+/an/	IDENTIO([LAB])/_V	MAXIO(C)/_V	*[LAB,C]	MAXIO(C)	IDENTIO([LAB])
☞ itan'dʒ̆reman			*		
itan'dʒ̆reØan		*!		*	
itan'dʒ̆renan	*!				*

3.7.1.2 *Second Member Nasal-Consonant-Initial Compounds*

In genitives where the possessor begins with a nasal, or reduplication where the reduplicating root begins and ends with a nasal, compounding creates a nasal + nasal cluster. In these situations the final nasal of the first half of the compound disappears—it is either deleted or merged into the initial nasal of the second half of the compound. A similar phenomenon occurs with nasal-final prefixes. The underlying form of the present active prefix seems to be /man/, on the basis of its appearance in vowel-initial stems, as in ['andʒ̆ran]—[man'andʒ̆ran]. When /man-/ is placed before a nasal-initial stem, however, the /n/ of /man-/ appears to have deleted or merged with the following nasal. The table in (82) shows some of the data for nasal + nasal clusters.

(82)

man- Prefixation		
Gloss	Unprefixed	Prefixed
bewitch	muˈsavi	ˌmamuˈsavi
regret	ˈnenin	maˈnenin
be bitter	ˈŋgidi	maŋˈgidi
cover	ˈmbumba	mamˈbumba
Reduplication		
Gloss	Root	Reduplicated
holy	ˈmasin	ˌmasiˈmasin

The pattern of these data is summarized in (83).

(83)

N1	N2	Result
n	m	m
n	n	n

I have chosen a coalescence analysis for these facts: the coalescence occurs (at the post-lexical level) because the ban on coalescing two consonants (84) and the ban on changing the place features of a consonant (85) are ranked below the ban on deleting consonants (86), the undominated ban on changing the feature [±NAS] (87), the undominated ban on a complex segment with two consonantal place features (88), and the undominated ban on consonant clusters wherein the second consonant is a nasal (89).

(84) IO-Uniformity(C):

Output a violation whenever two underlying consonants correspond to a single surface consonant (see §3.2.3.2).

(85) IdentIO(C-place):

Output a violation for any underlying consonant that corresponds with a surface segment that differs from it in consonantal place. Note—this is actually a simplification. Here, the real constraints that matter are IdentIO([-lab]), plus IdentIO([cor]).

(86) MaxIO(C):

Output a violation for each underlying consonant that does not correspond to a surface segment.

(87) IdentIO([±nas]):

Output a violation for any segment which has changed its specification for the feature [±nas].

(88) *Contour([αplace,C],[−αplace,C]):

Short form—*|[αplace,C][−αplace,C]|. *Output a violation wherever two consonant places abut within a single complex segment, as in [n͡m].* Note that this is actually an abbreviation for a number of constraints such as *Contour([cor,C],[dors,C]).

(89) *Sequence(C, [+nas]):

Short form—*C [+nas]. *Output a violation for each consonant+nasal sequence in a word.*

The resulting ranking is as follows:

(90) MaxIO(C), IdentIO([±nas]), *C [+nas], *|[αplace,C][−αplace,C]| ≫$_p$ IO-Uniformity(C), IdentIO(C-place)

as shown in (91).

(91) /masin+red/ → ['masin]$_s$+['masin]$_s$ → ['masin]$_w$+['masin]$_w$ → [ˌmasi'masin] 'holy+*red.*'

['masin]$_w$+['masin]$_w$	IdentIO([±nas])	*C [+nas]	*|[αplace,C][−αplace,C]|	MaxIO(C)	IdentIO(C-place)	IO-Uniformity(C)
☞ ˌmasi'masin					*	*
ˌmasiØ'masin				*!		
ˌmasi'nØasin				*!		
ˌmasi'n͡masin			*!			*
ˌmasim'masin		*!			*	
ˌmasin'masin		*!				
ˌmasim'basin	*!				*	

3.7.1.3 *Second Member Oral-Consonant-Initial Compounds*

The table in (92) exemplifies the general behavior of nasal-initial consonant clusters in Malagasy (the behaviors in the other contexts are simply special cases inside this one).

(92)

Gloss	Unreduplicated	Reduplicated
(a) *alive*	ˈvelun	ˌvelumˈbelun
(b) *cured*	ˈsitʃan	ˌsitʃanˈtsitʃan
(c) *quickly*	ˈha͡iŋgan	ˌha͡iŋganŋˈka͡iŋgan
(d) *first*	ˌvuaˈluhan	ˌvuaˌluhanˈduhan
(e) *average*	anˈtunin	anˌtuninˈtunin

The different types of clusters are shown in the following (consonant C_1 can be any nasal).

(93)

C_1	C_2	Cluster	C_1	C_2	Cluster
n,m,ŋ	p	mp	n,m,ŋ	z	nd͡z
n,m,ŋ	b	mb	n,m,ŋ	l	nd
n,m,ŋ	f	mp	n,m,ŋ	r	nd͡r
n,m,ŋ	v	mb	n,m,ŋ	ʧ	n̪ʧ
n,m,ŋ	t	nt	n,m,ŋ	k	ŋk
n,m,ŋ	d	nd	n,m,ŋ	g	ŋg
n,m,ŋ	s	nts	n,m,ŋ	h	ŋk

Remains a Cluster Notice here that the underlying cluster of two consonants remains a cluster on the surface. Nasal + oral clusters are the only consonant clusters allowed in the language—others are banned by the constraint *CC (94).

(94) *Sequence(C, C):

Output a violation for a surface consonant segment immediately following a surface consonant segment (so, for example, [abbbba] would incur three violations). Short form—*CC.

There are four ways to break up a consonant cluster: (1) insert a vowel between the two consonants, (2) delete one or both of the consonants, (3) change one of the consonants into a vowel, or (4) merge the two consonants into one. The first sort of fix would violate the constraint DepIO(V). The second fix, deletion, violates MaxIO(C). The third fix, changing a consonant into a vowel, violates the undominated constraint IdentIO([±syl]) (95).

(95) IdentIO([±syl]):

Output a violation for any surface segment that is consonantal while its underlying correspondent is vocalic, or vice versa.

Finally, the coalescence fix clearly violates IO-Uniformity(C), but since coalescence is desirable in some contexts (*e.g.*, see §3.7.1.2) that constraint cannot be the barrier in this case. Instead, coalescence is banned by the undominated constraints *|[αnas][−αnas]|, first seen in §3.3.2.1, and IdentIO([±nas]) (see §3.7.1.2). The final ranking therefore is (96), shown in tableau (97).

(96) DepIO(V), IdentIO([±syl]), *|[+nas][-nas]|, MaxIO(C), IdentIO([±nas]) $\gg_p$ *CC

(97) /lambu/ → [ˈlambu]$_s$ → [ˈlanbu]$_w$ → [ˈlambu] ‘wild boar’[21]

[ˈlanbu]$_w$	Ident-IO([±syl])	Ident-IO([±nas])	Max-IO([-nas,C])	*[+nas][-nas]	*[+son][-son]	Dep-IO(V)	Ident-IO([±low])	Max-IO(C)	*C C	Ident-IO([-lab])	*N V#	Ident-IO([cor])	Ident-IO([±cont])	*Lapse
☞ ˈlambu									*	*		*		
ˈlaØbu								*!						
ˈlanabu						*!								*
ˈlam͡bu				*!	*					*		*		
ˈlanØu			*!					*			*			
ˈlaabu	*!	*					*						*	*

Nasal Assimulation A second fact about these clusters is that the nasal assimilates in place to the following oral. This is analyzed as being the result of the general (inviolable) ban on heterogeneous consonant clusters (98). The resulting cluster takes on the place features of the second consonant because the second consonant is in a prevocalic position and therefore its place features are better preserved, via the undominated constraint Ident-IO([C-place])/_V (99) (see Steriade (2000 (in press)) for more argumentation in favor of this sort of constraint, albeit a surface-contextual form of it). The ranking is (100), as demonstrated in tableau (101).

(98) *Sequence([αplace,C], [−αplace,C], ∅):

[21] This derivation seems a bit odd, as an assimulated nasal becomes unassimilated, then assimilated again. The reason for this odd behavior (irrelevant as the behavior actually is to the point being illustrated here, which is that nasal + oral clusters survive as such) is that place assimilation in consonant sequences is kept undominated at the stem and post-lexical levels, but at the word level the simplest analysis for word-final nasal simplification was to ban all nasals other than [n], with the exception that [m] can survive prevocalically.

Output a violation for any consonant cluster in which the constituents differ in place features. This is actually an abbreviation for a fairly large set of constraints—*Sequence([lab, C], [dors, C]), *Sequence([cor, C], [lab, C]), etc. The short form is *[αplace, C] [−αplace, C].

(99) IdentIOBefore([C-place], V):

Output a violation for any surface consonant which corresponds to an underlying segment which differs from it in place features, if that underlying segment is followed in the underlying form by a vowel. Short form—IdentIO([C-place])/_V. Note that this constraint is actually an abbreviation of the constraints IdentIO([lab])/_V (77), IdentIO([cor])/_V, IdentIO([dors])/_V, and IdentIO([ant])/_V.

(100) IdentIO([C-place])/_V, *[αplace,C][−αplace,C] $\gg_p$ IdentIO([C-place])

(101) /lambu/ → ['lambu]$_s$ → ['lanbu]$_w$ → ['lambu] 'wild boar'

['lanbu]$_w$	IdentIO([+lab])/_V	IdentIO([-cor,-son])/_V	*[cor,C][lab,C]	*[dors,C][lab,C]	IdentIO([-lab])	IdentIO([cor])
☞ 'lambu					*	*
'laŋbu				*!		*
'lanbu			*!			
'landu	*!	*				

ALVEOLAR FRICATIVE HARDENING If the oral consonant in a nasal + oral consonant sequence is an alveolar fricative, it becomes the corresponding affricate (keeping its voice features), as in ['siʧan]:[ˌsiʧan'tsiʧan] (form (b) of table (92)). The consonant does not remain a fricative because an undominated stop assimilation constraint *[-CONT][+CONT,C] (102) outranks the violable identity constraints which protect fricatives (IDENTIO([+CONT,-SON])/_V (103) and IDENTIO([±CONT]) (104)).

(102) *SEQUENCE([-CONT], [+CONT,C], ∅):

Output a violation for each instance of a surface continuant consonant immediately preceded by a stop. Short form—*[-CONT][+CONT,C].

(103) IDENTIOBEFORE([+CONT,-SON], V):

Output a violation for any instance of an underlying fricative immediately followed in the underlying form by a vowel if that fricative corresponds to something which on the surface is not a fricative. For example, this would be violated by a change from /s/ to [t] or [ʦ], if /s/ is immediately followed by a vowel in the underlying form. This is as opposed to FEATMAXIO([+CONT, −SON])/_V (106), which would be violated by a change from /s/ to [t] but not [ʦ] because the features [+CONT, −SON] are preserved (FEATMAXIO([+CONT, −SON])/_V would also be violated if /s/ were deleted in this context, whereas IDENTIO([+CONT, −SON])/_V would not). Short form — IDENTIO([+CONT, −SON])/_V.

(104) IDENTIO([±CONT]):

Output a violation for any surface stop which is underlyingly a continuant, or vice versa.

The resulting ranking is thus:

(105) MaxIO(C), *[-cont][+cont,C] $\gg_p$ IdentIO([+cont,-son])/_V, IdentIO([±cont])

The fricative becomes an affricate and not a stop because the feature preserving constraint FeatMaxIO([+cont, −son])/_V (106) outranks the general ban on fricatives and affricates (*[+cont, −son], see (107)), the purpose of which will become clear when laterals are discussed later on. The ranking is in (108).

(106) FeatMaxIOBefore([+cont,-son], V):

Output a violation for any instance of an underlying phone of natural class [+cont, −son] *immediately followed in the underlying form by a vowel if the segment containing the phone* [+cont, −son] *corresponds to a complex segment (that is, a simple segment or a segment contour) on the surface that does not somewhere in it contain a phone from the natural class* [+cont, −son], *or if it corresponds to no surface segment at all*. See IdentIO-Before([+cont, −son], V) (103) for further explanation and examples. Short form—FeatMaxIO([+cont, −son])/_V.[22]

(107) *([+cont,-son]):

Output a violation for each surface member of the natural class [+cont,-son], *that is, for each surface fricative segment.*

(108) FeatMaxIO([+cont,-son])/_V $\gg_p$ *[+cont,-son]

[22]The fact that FeatMaxIO([+cont,-son])/_V was used here and not PtIdentIO([+cont,-son])/_V is primarily a matter of expediency—the two constraints are almost entirely equivalent, and no contextual faithfulness versions of the PtIdentIO family have yet been implemented in the program used to develop and test this analysis.

These rankings are illustrated in tableau (109) and further in (110), (114)–(116).

(109) /sasan+RED/ → [ˈsasan]$_s$+[ˈsasan]$_s$ → [ˈsasan]$_w$+[ˈsasan]$_w$ → [ˌsasanˈtsasan] 'some+*red.*'

[ˈsasan]$_w$+[ˈsasan]$_w$	*[-CONT][+CONT,C]	FEATMAXIO([+CONT,-SON])/_V	IDENTIO([+CONT,-SON])/_V	*[+CONT,-SON]	FEATMAXIO([+CONT,C])	IDENTIO([±CONT])
☞ ˌsasanˈtsasan			*	****		*
ˌsasanˈtasan		*!	*	***	*	*
ˌsasanˈsasan	*!			****		

LATERAL HARDENING If the oral consonant is /l/, it becomes [d]. This is accounted for by the above rankings, plus the undominated constraint *|[-LAT][+LAT]| (constraint (4) of §3.3.2.1), as shown in (110).

(110) /vualuhan+RED/ → [ˌvua'luhan]$_s$+['luhan]$_s$ → [ˌvua'luhan]$_w$+['luhan]$_w$ → [ˌvuaˌluhan'duhan]

'first+*red.*'

[ˌvua'luhan]$_w$+['luhan]$_w$	*[-CONT][+CONT,C]	*[-LAT][+LAT]	IDENTIO(±ɾ)	*[+CONT,-SON]	FEATMAXIO([+CONT,C])	IDENTIO([±CONT])
☞ ˌvuaˌluhan'duhan				***	*	*
ˌvuaˌluhan'dzuhan				****!		*
ˌvuaˌluhan'd͡ɾuhan			*!	***		*
ˌvuaˌluhan'd͡luhan		*!		***		*
ˌvuaˌluhan'luhan	*!			***		

The tableau also illustrates how *[+CONT,-SON] prevents /l/ from expressing its [+CONT] feature in an affricate, via the ranking in (111).

(111) *[+CONT,-SON] ≫$_p$ IDENTIO([±CONT]), FEATMAXIO([+CONT,C])

There is another way that /l/ could express [+CONT], and even [+SON] – it could become [ɾ], which is, after all, the closest phoneme in the language. Further, the sequence [d͡ɾ] is permitted in the language. However, the constraint IDENTIO(±ɾ) (112) prevents this change, ranked as in (113).

(112) IDENTIO(±[+SON,-LAT,-NAS,C]):

Short form—IDENTIO(±ɾ). *Output a violation for any underlying* [+SON,-LAT,-NAS,C] *segment that corresponds to something that is not* [+SON,-LAT,-NAS,C]. *Additionally, output a violation for any surface* [+SON,-LAT,-NAS,C] *segment that corresponds to something*

that is not [+SON,-LAT,-NAS,C]. This penalizes, for example, /l/→[ɾ] (because surface [ɾ] corresponds to a lateral) and /ɾ/→d͡ɾ (because underlying /ɾ/ corresponds to [d]). See §3.2.3.2 for more discussion of this sort of constraint.

(113) IDENTIO([±ɾ]) ≫$_p$ IDENTIO([±CONT]), FEATMAXIO([+CONT,C])

HARDENING OF RHOTICS If the oral consonant is /ɾ/, it becomes [d͡ɾ], by the ranking given in (105), adding IDENTIO([±ɾ]) to the bottom of that ranking. An example tableau appears in (114).

(114) /lavaraŋgan+RED/ → [ˌlavaˈraŋgan]$_s$+[ˈraŋgan]$_s$ → [ˌlavaˈrangan]$_w$+[ˈrangan]$_w$ → [ˌlavaˌraŋgan-ˈd͡raŋgan] 'veranda+*red.*'

[ˌlavaˈrangan]$_w$+[ˈrangan]$_w$	*[-CONT][+CONT,C]	IDENTIO(+/-ɾ)	FEATMAXIO([+CONT,C])	IDENTIO([±CONT])
☞ ˌlavaˌraŋganˈd͡raŋgan		*		*
ˌlavaˌraŋganˈdaŋgan		*	*!	*
ˌlavaˌraŋganˈraŋgan	*!			

HARDENING OF NON-ALVEOLAR FRICATIVES If the oral consonant is a peripheral fricative, it becomes the corresponding stop. This is due to the undominated constraint *|[−CONT] [+CONT, −COR]| (§3.3.2.1), as can be seen in tableaux (115) and (116).

(115) /velun+RED/ → [ˈvelun]$_s$+[ˈvelun]$_s$ → [ˈvelun]$_w$+[ˈvelun]$_w$ → [ˌvelumˈbelun] ‘alive+*red.*’

[ˈvelun]$_w$+[ˈvelun]$_w$	*[-CONT][+CONT,C]	*|[-CONT][+CONT,-COR]|	FEATMAXIO([+CONT,-SON])/_V	IDENTIO([+CONT,-SON])/_V	*[+CONT,-SON]	FEATMAXIO([+CONT,C])	IDENTIO([±CONT])
☞ ˌvelumˈbelun			*	*	*	*	*
ˌvelumˈb͡velun		*!		*	**		*
ˌvelumˈvelun	*!				**		

(116) /hâiŋgan+RED/ → [ˈhâiŋgan]$_s$+[ˈhâiŋgan]$_s$ → [ˈhâingan]$_w$+[ˈhâingan]$_w$ → [ˌhâiŋgaŋˈkâiŋgan] ‘quickly+*red.*’

[ˈhâingan]$_w$+[ˈhâingan]$_w$	IDENTIO([-COR,-SON])/_V	*[-CONT][+CONT,C]	*|[-CONT][+CONT,-COR]|	IDENTIO([-LAB])	IDENTIO([COR])	FEATMAXIO([+CONT,-SON])/_V	IDENTIO([+CONT,-SON])/_V	*[+CONT,-SON]	FEATMAXIO([+CONT,C])	IDENTIO([±CONT])
☞ ˌhâiŋgaŋˈkâiŋgan						***	*	*	*	*	*
ˌhâiŋgamˈpâiŋgan					*!*	***	*	*	*	*	*
ˌhâiŋgaŋˈkh͡âiŋgan			*!			***		*	**		*
ˌhâiŋgaŋˈhâiŋgan		*!				***			**		
ˌhâiŋganˈtâiŋgan	*!					**	*	*	*	*	*

Note in (116) that /h/ hardens to [k], a dorsal, rather than a coronal or a labial. This is because /h/ is more faithful to its property of not being labial (118) or coronal (117) than it

is to its property of not being dorsal.

(117) IDENTIO([-COR,-SON])/_V:

Output a violation for any underlying non-coronal obstruent that corresponds to a surface coronal and/or sonorant if said underlying non-coronal obstruent underlyingly precedes a vowel.

(118) IDENTIO([-LAB]):

Output a violation for any underlying non-labial segment that corresponds to a labial segment on the surface.

3.7.2 RELATED NASAL-INITIAL CLUSTER PHENOMENA

3.7.2.1 *Word-Initial Clusters*

A related data group is that of word-initial nasal clusters. In Malagasy words may begin with a nasal + oral consonant cluster, with the same data pattern as above, with the exception that no examples exist wherein the second consonant is voiceless. The table in (119) shows the behavior of word-initial consonant clusters beginning with a nasal. The underlying nasals in the examples with an initial voiceless consonant are Rich Base candidates, there just to show that such forms will not appear on the surface even if present in the underlying form (the chosen words happen to be spelled with an initial nasal for historical reasons—in an earlier stage of the language these words started with a vowel which has since disappeared),

(119)

Gloss	Hypothetical UR	Surface
numb from cold	/ŋgguli/	[ˈŋgguli]
cross-eyed	/ndʒula/	[ˈndʒula]
please	/mba/	[mba]
worker	/mpiasa/	[piˈasa]
ancestors	/ntaulu/	[ˈtaulu]

Essentially they work like this: if the second consonant is voiceless, the nasal is dropped, otherwise the cluster behaves as a medial cluster would. The primary motivating constraint here is *#[+NAS][-VCE] (120), which is undominated at the post-lexical level. The overall effect is due to ranking (121).

(120) *InitialSequence([+nas], [-vce]):

Output a violation for any word in which the first surface segment is a nasal and the second is voiceless. A specialization of the proposed universal *[+NAS][-VCE] (see Pater (1999)). Short form—*#[+NAS][-VCE].

(121) $$\text{IdentIO([-vce])/\_V}, *|[+\text{nas}][-\text{nas}]|, *\#[+\text{nas}][-\text{vce}], \text{DepIO(V)} \gg_p \text{MaxIO(C)},$$

where undominated MaxIO([-NAS,C]) causes the nasal to delete rather than the oral consonant. DepIO(V) prevents saving the underlying nasal by vowel insertion. IdentIO([-VCE])/_V (122) is another instance of increased faithfulness in underlying prevocalic position.

(122) IdentIOBefore([-vce], V):

Output a violation for any voiceless prevocalic segment in the underlying form that corresponds on the surface to a voiced segment. Short form—IDENTIO([-VCE])/_V.

All of this is illustrated in the following tableau:

(123) /ntâùlu/ → [nˈtâùlu]$_s$ → [nˈtâùlu]$_w$ → [ˈtâùlu] 'ancestors'

[nˈtâùlu]$_w$	MAXIO([-NAS,C])	*|[+NAS][-NAS]|	IDENTIO([-VCE])/_V	*#[+NAS][-VCE]	*|[+VCE][-VCE]|	*|[+SON][-SON]|	DEPIO(V)	MAXIO(C)	*C C	IDENTIO([±VCE])	IO-UNIFORMITY(C)
☞ ∅ˈtâùlu								*			
naˈtâùlu							*!				
nˈtâùlu				*!					*		
nˈdâùlu			*!						*	*	
ˈn͡tâùlu		*!			*	*					*
ˈnØâùlu	*!							*			

3.7.2.2 *The an- Prefix*

Like the word-initial nasal clusters of the previous section, the *an-* prefix (one of the language's active voice markers) introduces a nasal-initial cluster with slightly different properties from those described in §3.7.1.

Table 3.2 presents data that show the phonological effects caused by prefixation of the active prefix *an-* (also *man-, fan-, nan-,* but not, *e.g.,* locative *an-*)[23]. For the most part

[23]The data here is mostly from Paul (1995).

Gloss	Simple	Prefixed
Voiced Stops/Affricates		
(a) *bump*	ˈdun	manˈdun
(b) *squeeze*	ˈgedʑa	maŋˈgedʑa
(c) *stop*	dʑaˈnun	manˈdʑanun
(d) *grab*	ˈbeda	mamˈbeda
Voiced Continuants		
(e) *soaks*	ˈlun	manˈdun
(f) *progresses*	ˈrusu	manˈd͡rusu
(g) *measure*	ˈzehi	manˈdʑehi
Voiceless Consonants		
(h) *bite*	ˈka͡ikitʃa	maˈna͡ikitʃa
(i) *continue*	ˈtuhi	maˈnuhi
(j) *wash*	ˈsasa	maˈnasa
(k) *remove*	ˈtsa͡iŋguka	maˈna͡iŋguka
(l) *fence in*	ˈfefi	maˈmefi
(m) *put*	ˈpetʃaka	maˈmetʃaka
Unstable Cases		
(n) *cut*	ˈheti	maˈneti
(o) *ask*	ˈhataka	maŋˈgataka
(p) *help*	ˈvundʑi	maˈmundʑi
(q) *plant*	ˈvuli	mamˈbuli

Table 3.2: Nasal+Oral Clusters in *man-* Prefixation

these are the same effects caused by any underlying nasal+oral cluster except when the following consonant is voiceless, in which case the following consonant deletes, or is what would otherwise surface as [v] or [h], in which case the effect varies from root to root[24]. The pattern is schematized in the table below:

(124)

Stem-Initial C	Result	Stem-Initial C	Result
p	m	z	ndz
b	mb	l	nd
f	m	r	nd͡r
v	mb,m	ʧ	n
t	n	k	n
d	nd	g	ŋg
s	n	h	n,ŋg

Pater (1999) takes on similar data patterns, including this prefix, which shows up with similar patterns throughout the Western Austronesian languages. In this work he analyzes the situation as an instance of coalescence, where the final nasal of the prefix merges with the following consonant if it is nasal or voiceless. He also describes the traditional analysis for these phenomena:

> The standard analysis invokes two ordered rules to generate the single nasal from the underlying pair of segments: nasal assimilation, followed by a rule of root-initial, post-nasal, voiceless consonant deletion.

[24]There are also two words in which initial /b/ is dropped, but I have chosen to analyze these as listed forms.

For a variety of reasons a coalescence analysis is extremely difficult to integrate with the rest of my Malagasy analysis, and therefore I have followed the traditional line, implemented with sequentially-ordered levels rather than rules. However, the coalescence analysis may be a worthwhile area for future work. The effect of *an-* prefixation appears to be partially lexicalized, and as such I have moved part of its explanation from the postlexical level to the word level. At the word level voiceless following consonants are deleted, plus the consonants in forms that pattern like (n) and (p) in the table. Thus, the input to the postlexical level will be such that the normal nasal+oral cluster mechanisms there will do the right thing. The word level analysis works as follows.

Lexical Markings on the Prefix The nasal at the end of the *-an* prefix has a special lexical marking, here noted as "x," as in /n_x/. This passes through up to the start of the postlexical level, where the ranking in (125) removes it.

(125) *[+nas,+x], MaxIO(C) $\gg_p$ IdentIO([+nas,+x])

At the word level the ranking is reversed (126):

(126) MaxIO(C)/_V, MaxIO(C)/_C, IdentIO([+nas,+x]) $\gg_w$ *[+nas,+x]

The same is true of the stem level:

(127) MaxIO(C), IdentIO([+nas,+x]) $\gg_s$ *[+nas,+x].

Voiceless Consonants Following voiceless consonants are deleted, including underlying /h/, due to the ranking (129), as shown in (130). The constraint *[+nas,+x][-vce] (128) is a lexicalized reflex of Pater's (1999) universal *[+nas][-vce] constraint.

(128) *Sequence([+nas,+x], [-vce],):

Short form—*[+nas,+x][-vce]. *Output a violation for every instance of a lexically marked (+x) nasal followed by a voiceless segment.*

(129) MaxIO(C)/_C, IdentIO([-vce])/_V, *[+nas,+x][-vce] $\gg_w$ MaxIO(C)/_V

(130) /m+an$_{\chi}$+tuseh/ → /m/+[an$_{s}$ˈtuseh]$_{s}$ → [maˈn$_{\chi}$usik]$_{w}$ ‘*pres.*+*act.*+shove’

/m/+[an$_{\chi}$ˈtuseh]$_{s}$	MaxIO(C)/_C	IdentIO([-vce])/_V	*[+nas,+x][-vce]	MaxIO(C)/_V	[+nas,+x]	*[-nas,+ant,cor,C]	MaxIO(C)	*[+vce,-son]
☞ maˈn$_{\chi}$∅usik				*	*	*	*	
man$_{\chi}$ˈtusik			*!		*	**		
man$_{\chi}$ˈdusik		*!			*	**		*
ma∅ˈtusik	*!					**	*	

Voiced Labial Fricatives Following voiced labial fricatives are either deleted or changed to the corresponding stop. Which of these is chosen appears to be stem-specific, and not predictable from the isolation form of the stem. For example, the isolation form of the verb *to help* is [ˈvundzi] and the prefixed form is [maˈmundzi], so here the stem-initial segment is deleted. However the indistinguishable form [ˈvuli] ‘to plant’ behaves differently under prefixation—[mamˈbuli]. I have analyzed this situation as one where the underlying form of the initial consonant in [ˈvuli] differs from that of [ˈvundzi]. In particular, I claim that the underlying form of the latter is /vundzi/ whereas the underlying form of the former is /βuli/. This is of course, an abstract segment analysis, and therefore a bit controversial, so discussion of alternatives is in order. One alternative option would be to analyze one as underlying /v/, as before, and the other as underlying /b/. The problem there is that there is another behavior pattern that seems more likely to be due to underlying /b/: isolation [ˈbeda] ‘to grab’ becomes [mamˈbeda]. Another option would be /v/:/f/, which is ruled out by [ˈfefi]:[maˈmefi] ‘to fence in.’ The final option, of course, is to analyze the

alternations as being memorized on a stem-specific basis. I prefer the abstract segment analysis as exposing more regularity. Note that Pulleyblank (2003) argues for a similar approach in which abstract underlying representations involving segments not found in surface representations are licit just in the case where the abstract segments differ from non-abstract segments just in a feature or features that lead to surface contrast elsewhere in the language.

Having chosen somewhat abstract underlying forms, the analysis proceeds as follows. After the *an-* prefix, following voiced labial fricatives are changed or deleted, depending on whether or not they are [+STRID]. The sequences are not left alone because of an undominated constraint forbidding them:

(131) *SEQUENCE([+NAS,+X],[+CONT,-SON,LAB]):

Output a violation for any surface sequence of a lexically marked nasal segment followed by a labial fricative.

These labial fricatives do not change point of articulation because of the earlier-mentioned inviolable constraint IDENTIO([LAB])/_V (77). Underlyingly strident labial fricatives (/v/) resist change due the following undominated constraint:

(132) IDENTIO([+STRID]):

Assess a violation for any underlyingly strident segment that has a non-strident surface correspondent.

Because there is no acceptable labial strident fricative, these segments delete. Non-strident

labial fricatives, however, change to voiced stops because their barrier to change (133) is lower ranked. The ranking is as given in (134).

(133) IDENTIOIN([-STRID,+CONT,-SON,LAB], β):

Output a violation for any underlying /β/ that has a surface correspondent which is something other than a member of the natural class [-STRID,+CONT,-SON,LAB].

(134) *[+NAS,+X][+CONT,-SON,LAB], IDENTIO([+STRID]) $\gg_w$ MAXIO(C)/_V $\gg_w$ IDENTIOIN([-STRID,+CONT,-SON,LAB], β), IDENTIO([+CONT,-SON])/_V

See tableaux (135) and (136) for an illustration of this mechanism.

(135) /m+an$_x$+βule/ → /m/+[am$_x$'βule]$_s$ → [mam$_x$'buli]$_w$ '*pres.*+*act.*+plant'

/m/+[am$_x$'βule]$_s$	*[+NAS,+X][+CONT,-SON,LAB]	MAXIO(C)/_V	IDENTIOIN([-STRID,+CONT,-SON,LAB], β)	*[LAB,C]	MAXIO(C)	IDENTIO([+CONT,-SON])/_V	*[+CONT,-SON]	IDENTIO([±CONT])	IDENTIO([-STRID])	*[-COR,+VCE,+CONT,-SON]	*[+VCE,-SON]
☞ mam$_x$'buli			*	***		*		*			*
ma'm$_x$∅uli		*!		**	*						
mam$_x$'vuli	*!		*	***			*		*	*	*
mam$_x$'βuli	*!			***			*			*	*

(136) /m+an$_x$+vali/ → /m/+[am$_x$ˈvali]$_s$ → [maˈm$_x$ali]$_w$ '*pres.*+*act.*+answer'

/m/+[am$_x$vali]$_s$	IDENTIO([+STRID])	*[+NAS,+X][+CONT,-SON,LAB]	IDENTIOIN([+VCE,+CONT,-SON], [+STRID])	MAXIO(C)/_V	*[LAB,C]	MAXIO(C)	IDENTIO([+CONT,-SON])/_V	*[+CONT,-SON]	IDENTIO([±CONT])	*[-COR,+VCE,+CONT,-SON]	*[+VCE,-SON]
☞ ma$^{\prime}$m$_x$∅ali				*	**	*					
mam$_x$$^{\prime}$vali		*!			***			*		*	*
mam$_x$$^{\prime}$bali	*!		*		***		*		*		*

It still remains to be explained why /β/ appears as [v] on the surface. There will be some complications explored in §3.9.1.2, but essentially /β/ survives word-initially and post-nasally to become an input to the post-lexical level, and at that point it changes to [v] due to the agency of there-undominated *[-STRID,+CONT,-SON,LAB] (137), as shown in (138).

(137) *([-STRID,+CONT,-SON,LAB]):

Output a violation for each surface instance of a non-strident labial fricative.

(138) /βule+an/ → [βu'lean]$_s$ → [βu'len]$_w$ → [vu'len] 'plant+*pass.*'

[βu'len]$_w$	*[-strid,+cont,-son,lab]	FeatMaxIO([+cont,-son])/_V	IdentIO([+cont,-son])/_V	*[+cont,-son]	FeatMaxIO([+cont,C])	IdentIO([±cont])
☞ vu'len				*		
bu'len		*!	*		*	*
βu'len	*!			*		

In cases where the following consonant is deleted, a constraint preserving labial place in lexically marked nasals (139) outranks the general ban on labial consonants (140), and this keeps the nasal assimilated to the place of the deleted consonant. This place assimilation occurs at the stem level, with the same analysis (141) as at the post-lexical level.

(139) IdentIOIn([lab], [+nas,+x]):

Output a violation for any instance of an underlyingly labial lexically marked nasal with a surface correspondent that is not labial.

(140) IdentIOIn([lab], [+nas,+x]) $\gg_w$ *[lab,C]

(141) IdentIO(C-place)/_V, *[αC-place][−αC-place] $\gg_s$ IdentIO(C-place)

VELARS AND GLOTTALS It was stated above that underlying /h/ is deleted following the *an-* prefix. This accounts for the pattern [ˈheti]:[maˈneti] ('cut'), but leaves [ˈhataka]: [maŋˈgataka] unexplained. Here I have employed a similar analysis to that used for the voiced labial fricatives above—the underlying consonant in forms like [ˈhataka] is abstract /ɣ/. Underlying /ɣ/ surfaces as [h] in all contexts except following a nasal, because of a general (word-level undominated) ban on dorsal fricatives (142) and also the higher ranking of preservation of frication before a vowel with respect to the general ban on fricatives.

(142) *([+CONT,-SON,DORS]):

Output a violation for each instance of a surface dorsal fricative.

The relevant rankings are given in (143) and (144) and shown in (145).

(143) *[+CONT,-SON,DORS] $\gg_w$ IDENTIO([±VCE])/#_, IDENTIO([+VCE])/_V

(144) IDENTIO([+CONT,-SON])/_V $\gg_w$ *[+CONT,-SON]

(145) /ɣatah/ → [ˈɣatah]$_s$ → [ˈhatak]$_w$ 'ask'

[ˈɣatah]$_s$	*[+CONT,-SON,DORS]	IDENTIO([+CONT,-SON])/_V	IDENTIO([±VCE])/#_	*[+CONT,-SON]	IDENTIO([±CONT])	*[-COR,+VCE,+CONT,-SON]	IDENTIO([+VCE])/_V	*[+VCE,-SON]
☞ ˈhatak			*	*	*		*	
ˈgatak		*!			**			*
ˈɣatak	*!			*	*	*		*

After a nasal, /ɣ/ becomes [g] at the word level because of undominated *[+CONT,-SON,DORS] (142) and IDENTIO([+VCE])/[+NAS]_ (146) (also *[+NAS,+X][-VCE] (128), but /ɣ/ becomes [g] after lexically unmarked nasals as well), as illustrated in tableau (147).

(146) IDENTIOAFTER([+VCE], [+NAS])

Output a violation for any underlying sequence of a nasal followed by a voiced segment where that segment corresponds to a surface voiceless segment.

(147) /m+an$_x$+ɣatah/ → /m/+[aŋ$_x$ˈɣatah]$_s$ → [man$_x$ˈgatak]$_w$ '*pres.*+*act.*+ask'

/m/+[aŋ$_x$ˈɣatah]$_s$	IDENTIO([+VCE])/N_	*[+NAS,+X][-VCE]	*[+CONT,-SON,DORS]	MAXIO(C)/_V	MAXIO(C)	IDENTIO([+CONT,-SON])/_V	*[+CONT,-SON]	IDENTIO([±CONT])	*[-COR,+VCE,+CONT,-SON]	IDENTIO([+VCE])/_V	*[+VCE,-SON]
☞ man$_x$ˈgatak						*		**			*
maˈn$_x$∅atak				*!	*			*			
man$_x$ˈɣatak			*!				* *		*		*
man$_x$ˈhatak	*!	*					* *			*	

3.7.3 ORAL-CONSONANT-FINAL WEAK AND PSEUDO-WEAK ROOTS

Table 3.3 shows the behavior of oral-consonant-final weak and pseudo-weak roots in isolation vs. their behavior when suffixed. This behavior is summarized in Table 3.4; essentially, the isolation forms have fewer contrasting consonant phonemes than the suffixed forms do. My analysis for these phenomena is that the isolation forms are consonant-final at

	Present (*m*+stem)	Circumstantial (stem+*an*)	Gloss
(a)	maŋˈgataka	aŋgaˈtahan	*to ask for*
(b)	maˈnaraka	anaˈrahan	*to follow*
(c)	miˈanaʧa	iaˈnaran	*to study*
(d)	miˈsa͡uʧa	iˈsa͡uran	*to thank*
(e)	maˈnan͡draʧa	ananˈ͡dratan	*to promote*
(f)	mahafiˈnariʧa	ahafinaˈretan	*to please*
(g)	manˈ͡drakuʧa	an͡draˈkufan	*to cover*
(h)	maˈnahaka	anaˈhafan	*to scatter*
(i)	maˈnaluka	anaˈlufan	*to shade*
(j)	miˈlelaka	ileˈlafan	*to lick*

Table 3.3: Oral Consonant Simplification

the word level. At the word level these consonants simplify to either [ʧ], for coronals and some labials, or [k] for everything else. At the post-lexical level the vowel [a] is epenthesized word-finally, so both the isolation and suffixed forms end up having the stem-final consonant precede a vowel. The sections to follow give further details.

Before Epenthetic [a]	Elsewhere	Proposed UR
[ʧ]	[f]	[f]
[ʧ]	[t]	[t]
[ʧ]	[ʧ]	[ʧ]
[ʧ]	[ɾ]	[ɾ]
[k]	[f]	[ɸ]
[k]	[k]	[k]
[k]	[h]	[h]

Table 3.4: Word-Final Oral Consonant Summary

3.7.3.1 *Determination of Underlying Forms*

See Table 3.4 for a summary of the behavior of the final consonants in the oral-consonant-final stems. Ignoring the "Proposed UR" column for now, it appears that for the most part the isolation form of each root is predictable from the suffixed form. The only question then is what to do about the forms which are not completely predictable from the suffixed form. These are the forms where the suffixed stem ends in [f]. Here I have chosen a somewhat abstract analysis, analogous to those proposed in §3.7.2.2. The phoneme /f/ is quite similar to the bilabial voiceless fricative /ɸ/, differing only in the feature [±STRID]. As it happens, [ʧ] differs from [k] by this feature as well (also some others). Therefore I propose that the consonant that surfaces as [f] in the suffixed form and [ʧ] in the unsuffixed form is underlyingly /f/ and the other is /ɸ/.

3.7.3.2 *Word-Final Simplification*

The generalization here is that when the suffixed form of a weak stem ends in a dorsal or glottal consonant, the isolation form ends in [-ka]. When the suffixed form ends in a coronal consonant, the isolation form ends in [-ʧa]. If the suffixed form ends in a labial consonant, the isolation form is unpredictable—in §3.7.3.1 I proposed that labial stems that emerge as [-ka] are underlyingly non-strident, and those that emerge as [-ʧa] are underlyingly strident. Thus the prediction is that, although no weak stems appear to exist whose suffixed form ends in [b] or [p], the isolation form of these stems would be [-ka], if they did exist, since [b] and [p] are non-strident.

Velars As seen above, Malagasy seems to prefer alveolars when it comes to nasal consonants. Among oral consonants, however, there seems to be a preference for velars. Cross-linguistically this is quite unusual—alveolar oral consonants are generally preferred across languages, but Malagasy seems not to share this preference. A related result is that of Trigo (1988), who also found velars to be less marked than coronals in some languages, and Hume & Tserdanelis (2003), who found a language where the labial point of articulation seems to be the least marked. Perhaps there are available constraints to penalize every point of articulation, but in Malagasy the velar penalty is the lowest. Here the previously introduced constraint *[lab,c] (74) penalizes labial consonants while a new constraint (148) penalizes oral alveolar consonants.

(148) *[-nas,+ant,cor,C]:

Output a violation for each oral alveolar consonant in the surface representation.

An underlyingly word-final /k/ stays constant not because of identity constraints necessarily, but because there are no active[25] markedness constraints penalizing it:

(149) /fuk/ → [ˈfuk]$_s$ → [ˈfuk]$_w$ → [ˈfuka] 'absorb'

[fuk]$_s$	*[-NAS,+ANT,COR,C]	*[LAB,C]	MAXIO(C)	*[+CONT,-SON]	IDENTIO([+/-CONT])	IDENTIO([-STRID])
☞ ˈfuk		*		*		
ˈfutʃ		*		**!	*	*
ˈfuɸ		*	*!	*		
ˈfup		**!		*		
ˈfut	*!	*		*		

There is only one other non-labial underlying form that becomes [k] in word-final contexts, however, or at least only one for which there is evidence. That is the glottal fricative /h/, which surfaces as [k] word-finally because of the general ban on fricatives (places other than velar are not chosen because there are constraints against them).

(150) *[+CONT,-SON] $\gg_w$ IDENTIO([±CONT])

[25] "Active," when applied to a constraint, is used here to indicate that the constraint is not outranked by constraints that would cause it to have no candidate-selecting effect. That is, in the Optimality Theoretic generation process, which consists of a successive winnowing of a pool of candidate outputs, an active constraint is one that is ranked such that it will reduce the candidate pool for at least one potential input. An inactive constraint is ranked such that it never eliminates output candidates no matter what underlying form is being evaluated. For example, *[DORS] is inactive if it is outranked by IDENTIO([DORS]) and MAXIO([DORS]). In general I have not included inactive constraints in the analysis unless the constraints are active in some level of grammar and not in another.

The glottal fricative does appear in other contexts because of prevocalic faithfulness to fricatives (IDENTIO([+CONT,-SON])/_V (103)), which outranks the ban on fricatives:

(151) IDENTIO([+CONT,-SON])/_V $\gg_w$ *[+CONT,-SON].

These rankings are demonstrated by the tableaux (152) and (153).

(152) /pɛʧah/ → [ˈpɛʧah]$_s$ → [ˈpɛʧak]$_w$ → [ˈpɛʧaka] 'sit'

[ˈpɛʧah]$_s$	*[+CONT,-SON]	IDENTIO([±CONT])	IDENTIO([-STRID])
☞ ˈpɛʧak	*	*	
ˈpɛʧah	**!		
ˈpɛʧaʧ	**!	*	*

(153) /m+i+pɛʧah+a/ → /m/+[iˈpɛʧah]$_s$+/a/ → [mipeˈʧaha]$_w$ '*pres.*+*act.*+sit+*imp.*'

/m/+[iˈpɛʧah]$_s$+/a/	IDENTIO([+CONT,-SON])/_V	*[+CONT,-SON]	IDENTIO([±CONT])
☞ mipeˈʧaha		**	
mipeˈʧaka	*!	*	*

Final [k] is preferred to [g] in these words because of a general dispreference for voiced obstruents:

(154) *([+VCE,-SON]):

Output a violation for each voiced obstruent on the surface.

The selection of [k] in the case of /pɛʧah/ can be seen in the simplest tableau so far (155).

(155) /pɛʧah/ → [ˈpɛʧah]$_s$ → [ˈpɛʧak]$_w$ → [ˈpɛʧaka] 'sit'

[ˈpɛʧah]$_s$	*[+VCE,-SON]
☞ ˈpɛʧak	
ˈpɛʧag	*!

Beyond this, however, the constraint *[+VCE,-SON] has little effect (156)—it does not cause prevocalic voiced obstruents to change, nor does it cause consonant clusters involving voiced obstruents to simplify (158).

(156) IDENTIO([+VCE])/_V, IDENTIO([±CONT]) $\gg_w$ *[+VCE,-SON]

(157) /atiduha/ → [ˌatiˈduha]$_s$ → [ˌatiˈduha]$_w$ 'brain'

[ˌatiˈduha]$_s$	IDENTIO([+VCE])/_V	*[+VCE,-SON]
☞ ˌatiˈduha		*
ˌatiˈtuha	*!	

(158) /in$\widehat{\text{dr}}$aɪ/ → [in$^{\text{ˈ}}\widehat{\text{dr}}$aɪ]$_s$ → [in$^{\text{ˈ}}\widehat{\text{dr}}$aɪ]$_w$ ‘again’

[in$^{\text{ˈ}}\widehat{\text{dr}}$aɪ]$_s$	IDENTIO([±CONT])	*[+VCE,-SON]
☞ in$^{\text{ˈ}}\widehat{\text{dr}}$aɪ		*
inˈraɪ	*!	

ALVEOLARS No alveolars survive as such in final position, although non-fricative alveolars survive as the post-alveolar affricate [ʧ]. Word-final alveolars can become [ʧ] because the cross-linguistically unusual constraint *[-NAS,+ANT,COR,C] (148) banning alveolar oral consonants outranks the constraints which penalize [ʧ] and the feature changes that are necessary to change an alveolar stop to a post-alveolar affricate (see ranking (159)).

(159) *[-NAS,+ANT,COR,C] $\gg_w$ *[+CONT,-SON], IDENTIO([±CONT]), IDENTIO([-STRID]).

Underlying alveolars do not become velars because the ban on changing coronals into non-coronals (or deleting them) outranks the penalizing constraints:

(160) IDENTIO([COR]), MAXIO(C) $\gg_w$ *[+CONT,-SON], IDENTIO([±CONT]), IDENTIO([-STRID]).

The effect of these rankings is illustrated in the following tableau:

(161) /avut/ → ['avut]$_s$ → ['avutʃ]$_w$ → ['avutʃa] 'redeem (redemption)'

[avut]$_s$	IDENTIO([COR])	*[-NAS,+ANT,COR,C]	MAXIO(C)	*[+CONT,-SON]	IDENTIO([±CONT])	IDENTIO([-STRID])
☞ 'avutʃ				**	*	*
'avuØ			*!	*		
'avut		*!		*		
'avuk	*!			*		

This change does not occur before a vowel because of a prevocalic positional faithfulness constraint (162), in ranking (103), shown in tableau (164).

(162) IDENTIOBEFORE([+ANT],V):

Short form—IDENTIO([+ANT])/_V. *Output a violation for each instance of the underlying sequence of an anterior consonant preceding a vowel where the anterior consonant corresponds to a surface segment that is not a member of the natural class* [+ANT]. For example, [atʃa] derived from underlying /ata/ would violate the constraint once.

(163) IDENTIO([+ANT])/_V $\gg_w$ *[-NAS,+ANT,COR,C].

(164) /avut+an/ → [aˈvutan]$_s$ → [aˈvutan]$_w$ 'redeem+*pass.*'

[aˈvutan]$_s$	IDENTIO([+ANT])/ _ V	*[-NAS,+ANT,COR,C]	*[+CONT,-SON]	IDENTIO([±CONT])	IDENTIO([-STRID])
☞ aˈvutan		*	*		
aˈvutʃan	*!		**	*	*

LABIALS As opposed to the alveolars, the only evidence for underlying labials in word-final position seems to come from fricatives[26]. For weak stems with a proposed underlying labial final consonant, there are two different patterns. In one pattern, forms with a vowel-initial suffix show the segment [f] whereas the unsuffixed form has [k]. In the other, the suffixed form has [f] but the unsuffixed form has [tʃ]. As mentioned in §3.7.3.1, I am analyzing the underlying form behind the first pattern as /ɸ/ and the second as /f/ (I also propose underlying voiced labial fricatives in some consonant-final stems, as will be seen in §3.9). For underlying /ɸ/, the ranking in (165) propels a change away from the labial place of articulation and keeps the consonant from being deleted entirely in the unsuffixed form, and the presence of *[+CONT,-SON] in the grammar causes [k] to be preferred over [tʃ], as shown in tableaux (166) and (167).

(165) MAXIO(C), *[LAB,C] $\gg_w$ IDENTIO([±CONT]), IDENTIO([LAB])

[26] I can find no instances of [b] or [p] in consonant-final stems. As previously stated, the prediction of this analysis is that if such stems were found, /b#/ and /p#/ would surface as [ka].

(166) /m+i+lelaɸ/ → /m/+[iˈlelaɸ]$_s$ → [miˈlelak]$_w$ '*pres.*+*act.*+lick'

/m/+[iˈlelaɸ]$_s$	*[LAB,C]	MAXIO(C)	*[+CONT,-SON]	IDENTIO([±CONT])	IDENTIO([-STRID])	IDENTIO([LAB])
☞ miˈlelak	*			*		*
miˈlelaʧ	*		*!	*	*	*
miˈlelaØ	*	*!				
miˈlelaɸ	**!		*			

(167) /i+lelaɸ+an/ → [iˈlelaɸ]$_s$+/an/ → [ileˈlaɸan]$_w$ '*act.*+lick+*circum.*'[27]

/[iˈlelaɸ]$_s$+/an/	IDENTIO([LAB])/_V	*[LAB,C]	IDENTIO([+CONT,-SON])	*[+CONT,-SON]	IDENTIO([±CONT])	IDENTIO([-STRID])	IDENTIO([LAB])
☞ ileˈlaɸan		*		*			
ileˈlafan		*		*		*!	
ileˈlakan	*!		*		*		*

The underlying labial remains labial in the suffixed form because of the already-seen ranking

(168) IDENTIO([LAB])/_V ≫$_w$ *[LAB,C].

[27] Here the surface form becomes [ileˈlafan] due to post-lexical dispreference for non-strident fricatives.

Note that lexical-level [ɸ] becomes [f] at the post-lexical level due to the undominated constraint *[−STRID, +CONT, −SON, LAB], which outranks IDENTIO([±STRID]). In fact, IDENTIO([±STRID]) is low-enough ranked that it is inactive at the post-lexical level.

Similar forces work on underlying /f/, but the ranking

(169) IDENTIO([+STRID]) $\gg_w$ *[+CONT,-SON]

causes the alternative [ʧ] to be chosen, as shown in the following tableau:

(170) /rakuf/ → ['rakuf]$_s$ → ['rakuʧ]$_w$ 'cover'

[rakuf]$_s$	IDENTIO([+STRID])	*[LAB,C]	MAXIO(C)	*[+CONT,-SON]	IDENTIO([±CONT])	IDENTIO([LAB])
☞ 'rakuʧ				*	*	*
'rakuØ			*!			
'rakuf		*!		*		
'rakuk	*!				*	*

3.7.3.3 *Second Member Nasal-Consonant-Initial Compounds*

Genitives where the object of possession is an oral-consonant-final weak root and the possessor begins with a vowel introduce oral + nasal clusters. In the output, the nasal disappears and the oral consonant is preserved.

(171)

Gloss	Object	Possessor	Genitive
chicken's foot	ˈtuŋgutʃa	aˈkuhu	ˌtuŋgutʃaˈkuhu
Soa's shoulder	ˈsuruka	iˈsua	ˌsurukiˈsua

From (173) it appears that this behavior derives from the combination of a number of undominated constraints. The essential characterization is that an underlying nasal consonant deletes after an underlying oral consonant. This happens for the following reasons:

1. Nasality does not change, due to IDENTIO([±NAS]) (87).
2. Nasal-final clusters are banned, due to *C [+NAS] (89).
3. Clusters may not simplify by becoming a single complex segment with differing values for [±NAS], due to *|[αNAS][−αNAS]| (6).
4. Oral consonants may not be deleted (172).

(172) MAXIO([-NAS,C]):

Output a violation for each underlying oral consonant that does not appear on the surface.

(173) /tuŋgut/+/n/+/akuhu/ → [ˈtuŋgut]$_s$+/n/+[aˈkuhu]$_s$ → [ˈtuŋgu̲t̲n]$_w$+[aˈkuhu]$_w$ →

[ˌtuŋgutʃaˈkuhu] ‘foot+*gen.*+chicken=chicken foot’

[ˈtuŋgu̲t̲n]$_w$+[aˈkuhu]$_w$	IDENTIO([±NAS])	MAXIO([-NAS,C])	*|[-NAS][+NAS]|	*C [+NAS]	IDENTIO([+VCE])/_V	*|[-VCE][+VCE]|	MAXIO(C)	*C C	IDENTIO([±ANT])	IDENTIO([±VCE])	*[+FRIC]	IDENTIO([±CONT])	IO-UNIFORMITY(C)
☞ tˌuŋgutʃØakˈuhu							*	*			**	*	
tˌuŋgutnakˈuhu				*!				**	*		*		
tˌuŋgu͡tnakˈuhu			*!			*		*	*		*		*
tˌuŋguØnakˈuhu		*!					*	*			*		
tˌuŋgutakˈuhu	*!				*			*	*	*	*		*

3.7.3.4 *Second Member Oral-Consonant-Initial Compounds*

The behavior of oral+oral clusters, shown in Table 3.5, is almost identical to that of nasal+oral clusters. The only difference is that the output is a single consonant, formed by coalescence rather than deletion. The remaining consonant keeps the voicing and place of articulation features of the second of the two underlying consonants, but becomes an obstruent stop.

The explanation for why the consonants simplify by coalescence rather than deletion comes from ranking (174), as illustrated in (175).

(174) MAXIO(C), *CC $\gg_p$ IO-UNIFORMITY(C)

Gloss	Isolation Form	Hyp. Word-Output	Reduplicated
Second Consonant is a Stop or Affricate			
offer	ˈtulutʃa	[ˈtulutʃ]$_w$+[ˈtulutʃ]$_w$	ˌtuluˈtulutʃa
cut	ˈtapaka	[ˈtapak]$_w$+[ˈtapak]$_w$	ˌtapaˈtapaka
Second Consonant is an Alveolar Fricative			
writing	ˈsuratʃa	[ˈsuratʃ]$_w$+[ˈsuratʃ]$_w$	ˌsuraˈtsuratʃa
thing	ˈzavatʃa	[ˈzavatʃ]$_w$+[ˈzavatʃ]$_w$	ˌzavaˈd͡zavatʃa
Second Consonant is a Peripheral Fricative			
known	ˈfantatʃa	[ˈfantatʃ]$_w$+[ˈfantatʃ]$_w$	ˌfantaˈpantatʃa
fault	ˈheluka	[ˈheluk]$_w$+[ˈheluk]$_w$	ˌheluˈkeluka
selling	ˌvarutʃa	[ˈvarutʃ]$_w$+[ˈvarutʃ]$_w$	ˌvaruˈbarutʃa
Second Consonant is a Sonorant			
flexibility	ˈlefaka	[ˈlefak]$_w$+[ˈlefak]$_w$	ˌlefaˈdefaka
conversation	ˈɾesaka	[ˈɾesak]$_w$+[ˈɾesak]$_w$	ˌɾesaˈd͡ɾesaka

Table 3.5: Oral+Oral Consonant Clusters

(175) /tulur+RED/ → [ˌtulurˈtulur]$_s$ → [ˌtulut̲ˈtulutʃ]$_w$ → [ˌtuluˈtulutʃa] 'offer+*red.*'

[ˌtulut̲ˈtulutʃ]$_w$	MaxIO([-nas,C])	MaxIO(C)	*C C	IdentIO([±ant])	*[+cont,-son]	IdentIO([±cont])	IO-Uniformity(C)
☞ ˌtuluˈtulutʃa				*	*		*
ˌtulutˈtulutʃa			*!	*	*		
ˌtuluØˈtulutʃa	*!	*			*		
ˌtuluˈʧØulutʃa	*!	*			**	*	

Some of the forms, such as ['fantatʃa]:[fanˈtarin]:[ˌfantaˈpantatʃa], would be paradoxical in a mono-stratal analysis. In [ˌfantaˈpantatʃa] an underlying continuant merges with another underlying continuant, but the result is a stop. Where the first position consonant is not a fricative, it makes sense that its non-continuant nature is preserved in the merged consonant, via the independently needed undominated constraint PtIdentIO([-son,-cont]) (176) (it is independently needed so that underlying affricates remain as such on the surface). Under the lexical/post-lexical analysis, however, the first position consonant is always a stop or affricate at the level (post-lexical) where the coalescence occurs, due to word-final consonant neutralization.

(176) PtIdentIO([-son,-cont]):

Assess a violation for any surface segment that corresponds to an underlying obstruent stop and does not have the feature [-son,-cont] *at any point within its segment contour.*

(177) /surat+RED/ → [ˈsurat]$_s$+[ˈsurat]$_s$ → [ˈsuratʃ]$_w$+[ˈsuratʃ]$_w$ → [ˌsuraˈtsuratʃa] ‘writing+*red.*’

[ˈsuratʃ]$_w$+[ˈsuratʃ]$_w$	PtIdentIO([-son,-cont])	FeatMaxIO([+cont,-son])/_V	IdentIO([+cont,-son])/_V	*[+cont,-son]	FeatMaxIO([+cont,Cl])	IdentIO([±cont])
☞ ˌsuraˈtsuratʃa				***	*	*
ˌsuraˈturatʃa		*!	*	**	**	**
ˌsuraˈsuratʃa	*!			***		*

When consonant clusters coalesce, they take on the place and voicing value of the second consonant, due to the following ranking:

(178) IdentIO([C-place])/_V, IdentIO([±voice])/_V, *CC ≫$_p$ IdentIO([C-place]), IdentIO([±voice])

This behavior is partially illustrated in tableaux (179) and (180).

(179) /surat+RED/ → [ˈsurat]$_s$+[ˈsurat]$_s$ → [ˈsuratʃ]$_w$+[ˈsuratʃ]$_w$ → [ˌsuraˈtsuratʃa] ‘writing+*red.*’

[ˈsuratʃ]$_w$+[ˈsuratʃ]$_w$	*C C	IdentIO([±ant])	*[+cont,-son]	FeatMaxIO([+cont,C])	IdentIO([±cont])	IO-Uniformity(C)
☞ ˌsuraˈtsuratʃa		*	***	*	*	*
ˌsuraˈʧsuratʃa	*!		****			

(180) /varut/+RED → [ˈvarut]$_s$+[ˈvarut]$_s$ → [ˈvarʊʃ]$_w$+[ˈvarʊʃ]$_w$ → [ˌvaruˈbarʊʃa] ‘selling+*red.*’

[ˈvarʊʃ]$_w$+[ˈvarʊʃ]$_w$	*|[-VCE][+VCE]|	*C C	IDENTIO([±VCE])	IO-UNIFORMITY(C)
☞ ˌvaruˈbarʊʃa			*	*
ˌvarupˈbarʊʃa		*!		
ˌvaruˈp͡barʊʃa	*!			*

All the other behaviors shown in Table 3.5 fall out of the same constraint rankings described in §3.7.1.3.

3.7.4 RANKINGS SO FAR

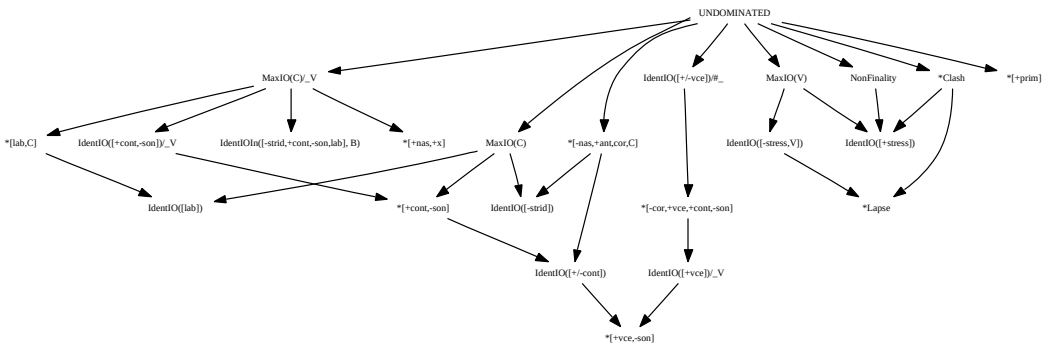
3.7.4.1 *Stem Level*

At the stem level the only ranking facts that have been established so far are the violable status of IDENTIO([CPLACE]) and the inviolable status of IDENTIO([CPLACE])/_V, *[αCPLACE] [−αCPLACE], and the constraints listed in 3.5.5 as being undominated in the other levels.

3.7.4.2 *Word Level*

Other than the generally undominated constraints, the following constraints have been established as undominated at the word level: IDENTIO([LAB])/_V and IDENTIO([±SYL]).

The rankings of violable constraints are as follows, up to this point:

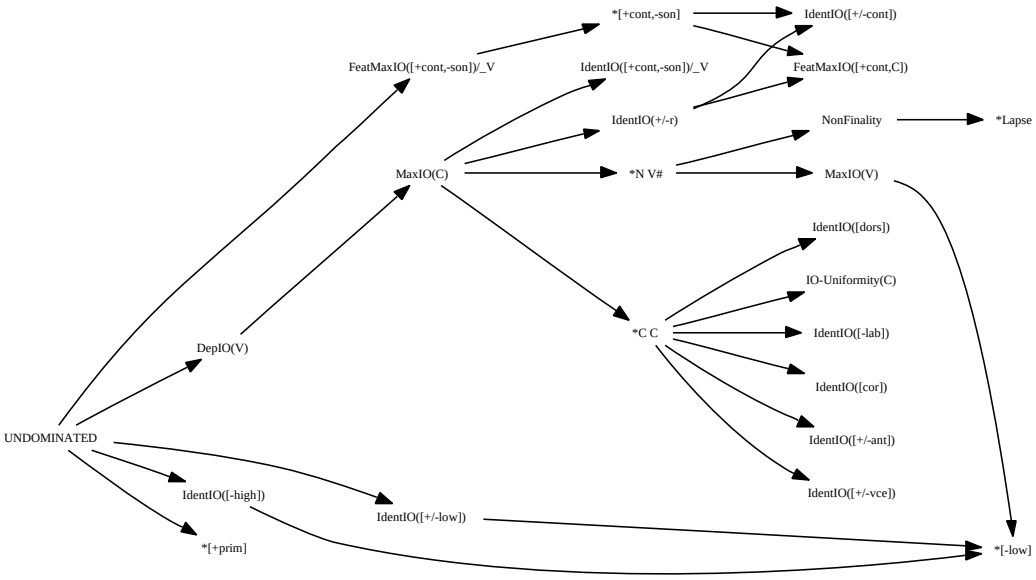


3.7.4.3 *Post-Lexical Level*

Other than the generally undominated constraints, the following constraints have been established so far as undominated at the post-lexical level: DEPIO(C), MAXIO([+STRESS]),

MaxIO([−nas, C]), IdentIO([±nas]), *C [+nas], IdentIO([±syl]), *[αC-place] [−αC-place], *[−cont] [+cont, C], and PtIdentIO([−son, −cont]).

The rankings of violable constraints are as follows, up to this point:



3.8 EXCEPTIONS TO THE BASIC STRESS PATTERN

This section details various phenomena related to stress in Malagasy, including opaque stress under vowel loss, quantity sensitive stress, and phonemic stress.

3.8.1 HIATUS AND OPAQUE STRESS

There seems to be a pattern of opaque stress under vowel loss. In certain instances of created vowel hiatus, one of the vowels deletes, and the resulting vowel generally carries stress. In (181) group (a) does not exhibit this pattern—falling patterns (high to low) are

apparently acceptable as hiatuses in Malagasy.

(181)

	Gloss	Less Morph.	More Morph.
(a)	*wait*	ˈandr͡i	miˈandr͡i
		/andr͡i/	/m+i+andr͡i/
	believes	ˈmin	miˈnua
		/m+inu/	/m+inu+a/
	understood	ˈazu	ˌazuˈazu
		/azu/	/azu+azu/
(b)	*work*	miˈasa	iaˈsan
		/m+i+asa/	/i+asa+an/
	reap	miˈʣinʣa	iʣinˈʣan
		/m+i+ʣinʣa/	/i+ʣinʣa+an/
	carry, bring	miˈtundr͡a	ˌitunˈdr͡an
		/m+i+tundr͡a/	/i+tundr͡a+an/
	change	ˈuva	ˌuvˈuva
		/uva/	/uva+uva/
(c)	*look at*	miˈʣeri	iʣeˈren
		/m+i+ʣere/	/i+ʣere+an/
	back-carry	ˈbabi	baˈben
		/babe/	/babe+an/
	confess	miˈâiki	iˌâiˈken
		/m+i+âike/	/i+âike+an/
(d)	*spit*	ˈivi	ˌivˈivi
		/ivi/	/ivi+ivi/

In group (b) the affixed vowel /a/ cannot follow the stem-final vowel /a/ and opaque stress can be seen in the resulting word-final main stress. Group (c) is much the same, with stem-final /e/ similarly incompatible with affixed /a/. Finally, group (d) partially illustrates a general pattern wherein adjacent identical vowels merge. Note that some of this data (mostly in (c)) shows evidence of vowel weakening, where unstressed word-final front vowels raise to [i]. This phenomenon is analyzed in §3.10.

When one of two adjacent vowels delete and one carries stress, the resulting vowel must carry stress as well, with one exception to be noted later. The first part of this, of course, is the vowel deletion—why does it occur? Looking at Table (181), it is possible to see that the language appears to allow some vowel combinations in hiatus and disallow others. The analysis for vowel deletion here consists first of a set of constraints that bans all of the vowel combinations that are never found in Malagasy:

(182) *Sequence(V_i, V_i, ∅):

Short form—*V_iV_i[28].

(183) *Sequence(a, V, ∅):

Short form—*aV. *Output a violation for any low vowel followed immediately by another vowel.*

(184) *Sequence(e, a, ∅):

Short form—*ea.

[28]Implemented as a set of constraints such as *Sequence(i, i, ∅), *Sequence(e, e, ∅), one for each vowel type.

(185) *Sequence(e, i, ∅):

Short form—*ei.

I will abbreviate these collectively as *BadVSeq. My analysis for this is that when a banned vowel sequence occurs, one of the vowels is deleted, unless for some reason both vowels carry stress. If only one of the vowels carries stress, the unstressed vowel is deleted. If both vowels are unstressed, the second vowel deletes[29]. One of the rankings that account for this is (186). Note also in this ranking that contexts for positional faithfulness with respect to the Max family of constraints are presumed to be underlying, thus the second "V" in MaxIO(V)/_V represents an underlying vowel. The ranking is illustrated in tableaux (187)–(189).

(186) *BadVSeq, IdentIO([±round]), IdentIO([±back]) $\gg_w$ MaxIO(V), MaxIO(V)/_V, *Clash, NonFinality

(187) /i+asa+an/ → $[\text{i}'\text{asa}]_s$+/an/ → $[\text{ia}'\text{san}]_w$ '*act.*+work+*circum.*'

$[\text{i}'\text{asa}]_s$+/an/	*BadVSeq	NonFinality	MaxIO(V)
☞ ia'saØn		*	*
ia'saan	*!		

[29] I have not thoroughly investigated prefix-stem boundary cases. Perhaps MaxIO(V)/_V actually reflects increased faithfulness to stem vowels.

(188) /ivi+RED/ → [ˌivi'ivi]$_s$ → [ˌi'vivi]$_w$ 'spit+*red.*'

[ˌivi'ivi]$_s$	MaxIO([+stress])	IdentIO([±back])	IdentIO([±round])	IdentIO([-low])/V_	*BadVSeq	MaxIO(V)/_V	MaxIO(V)	IdentIO([-stress,V])	*Clash
☞ ˌivØ'ivi						*	*		*
ˌivi'ivi					*!				
ˌivi'uvi		*!	*						
ˌiv'iØvi	*!			*			*	*	*

(189) /i+ʤere+an/ → [i'ʤere]$_s$+/an/ → [iʤe'ren]$_w$ '*act.*+look at+*circum.*'

[i'ʤere]$_s$+/an/	IdentIO([±round])	*BadVSeq	NonFinality	MaxIO(V)	*e C_0#	IdentIO([-high])	IdentIO([+low])
☞ iʤe'reØn			*	*	*		
iʤe'rean		*!					
iʤe'reun	*!					*	*

The other is (190), illustrated in tableau (188).

(190) MaxIO([+stress]) $\gg_w$ MaxIO(V)/_V

The basic statement of the opaque stress pattern here is that stress is attracted to a vowel just before another deleted one. This could be something related to compensatory lengthening, except that there appears to be no lengthening. Alternatively, it could be related to

stress inheritance, except that it occurs in forms such as [iˈʤere]$_s$+/an/ → [iʤeˈren]$_w$ where neither the surviving vowel ([e], here) or the deleted vowel ([a], here) has a previously assigned stress. A fairly direct encoding of the basic statement above is used (complicated somewhat by a restriction to cases where the deleted vowel was underlyingly [+BACK]):

(191) *SEQUENCEIO([V,-STRESS], V, ZERO, [+BACK], ∅):

Short form—OPQSTRESS. *Output a violation for any unstressed vowel appearing directly before a deleted back vowel (opaque stress attraction appears not to apply when the second vowel is a front vowel, as will be seen in §3.10.2).* Reminder: *SEQUENCEIO(S_1, U_1, S_2, U_2, I) signifies *S_1 : U_1 I_0 S_2 : U_2, and ZERO indicates a deleted element, so a literal reading of the constraint would be "output a violation for any two-segment sequence wherein the first segment is a non-stressed vowel that is also a vowel underlyingly and the second is a deleted segment where the underlying form is a member of the natural class [+BACK]."

The following ranking is also necessary (illustrated in (193)):

(192) MAXIO(V)/_V, OPQSTRESS $\gg_w$ IDENTIO([-STRESS,V]), NONFINALITY, IDENTIO([+STRESS]), *LAPSE

(193) /m+i+sala+RED+a/ → /m/+[i˛sala'sala]$_s$+/a/ → [mi˛salasa'la]$_w$ '*pres.*+*act.*+hesitate+*imp.*'

/m/+[i˛sala'sala]$_s$+/a/	OpqStress	MaxIO(V)/_V	NonFinality	IdentIO([-stress,V])	*Lapse	IdentIO([+stress])
☞ mi˛salasa'laØ			*	*	*	*
mi˛sala'salØa		*!				
mi˛sala'salaØ	*!					

3.8.2 QUANTITY-SENSITIVE STRESS

Recall from §3.4.1 that the basic stress pattern of Malagasy is right-to-left trochaic, enforced by *Clash, *Lapse, and NonFinality, each unviolated as far as the basic pattern is concerned. However, the data in (194)[30] shows that stress is attracted to diphthongs in Malagasy, and this quantity-sensitive stress attraction can cause violations of two of these basic constraints.

[30]Note that the stem level output for the form [i˛âi'ken] is [i˛âi'kean]$_s$. I have analyzed the underlying form as /i+âike+an/, where all affixes are stem level. Thus, the final-syllable stress on this form is due to the opaque stress phenomenon analyzed in §3.8.1.

(194)

Gloss	Form
(a) *command*	ˈbai̯ku
(b) *to do*	maˈnau̯
(c) *gnat*	aˈlui̯
(d) *patched*	ˌbemiˈrai̯
(e) *since*	ˌhatʃiˈzai̯
(f) *give in*	iˌai̯ˈken
(g) *darkened*	ˌai̯ˈzinin

In particular, in forms (b)–(e) stress attraction to diphthongs leads to final stress, and forms (f) and (g) exhibit stress clash. The attraction of stress to diphthongs comes from the Weight to Stress Principle (WSP), encoded here as follows.

(195) *Contour([V,αVplace,-stress],[V,−αVplace,-stress]):

Short form — WSP. *Output a violation for each diphthong that does not carry stress.* The constraint is formulated in this way because in the present representation there is no way to distinguish separate elements in a contour unless they differ on some feature[31].

In forms such as [aˈlui̯] and [maˈnau̯], WSP causes stress to gravitate to the final syllable in contravention of the NonFinality constraint. This could be repaired by splitting the diphthong into separate syllables, but that would violate IO-Integrity(V) (196).

[31] In the implementation there is actually no direct encoding for feature variables such as α, so this abstract constraint is represented by a bank of constraints, one for each matching featural specification. Only two are actually necessary in Malagasy—*|[V,-high,-stress][V,+high,-stress]| and *|[V,+back,-stress][V,-back,-stress]|.

(196) IO-Integrity(V):

An underlying vocalic segment (whether a simple vowel or a diphthong) may not correspond to more than one surface segment. Output a violation for each corresponding segment other than the permitted first one, e.g.*, if an underlying vowel corresponds to three surface vowels, output two violations.*

The diphthong is in actuality not split. This indicates that WSP and IO-Integrity(V) outrank NonFinality. See (198) for an illustration.

(197) WSP, IO-Integrity(V) ≫ NonFinality (applies at all levels)

(198) /alûi/ → [aˈlûi]$_s$ 'gnat'

/alûi/	WSP	IO-Integrity(V)	NonFinality
☞ aˈlûi			*
aˈlui		*!	
ˈalûi	*!		

As seen in (f) and (g) above, forms with diphthongs can exhibit stress clash. This clash is not broken up by vowel insertion, so DepIO(V) is active here. Further, these never exhibit stress lapse in the final two syllables of a word, due to the following constraint:

(199) *FinalSequence([V,-stress], [V,-stress], C):

(Short form — *FinalLapse) *Output a violation for every word-final lapse.*

Thus, as seen in (201), the following ranking must be in effect:

(200) *FinalLapse, NonFinality, DepIO(V) ≫ *Clash (at all levels),

(201) /a͡izin+in/→[ˌa͡iˈzinin]$_s$ 'darkness+*pass.*'

[ˈa͡izin]$_s$+/in/	DepIO(V)	*FinalLapse	NonFinality	*Clash	*Lapse
☞ ˌa͡iˈzinin$_s$				*	
ˌa͡iziˈnin$_s$			*!		
ˈa͡izinin$_s$		*!			*
ˌa͡izaˈinin$_s$	*!				

3.8.3 PHONEMIC STRESS

Other than some stress-bearing prefixes, which I will not go into other than to say that they are added at the word or post-lexical level, depending on the prefix, the only evidence of phonemic stress in the language seems to involve the final syllable. In Malagasy occasionally stress appears on the final syllable of a word even if that syllable does not include a diphthong. In many of these instances the stressed final syllable takes the vocalic value [e] (not all, however—for example, there is the word *vuˈvu*). Pearson (1994) has taken this as evidence that the phoneme /e/ is inherently long, but it appears that underlying /e/ can appear on the surface without stress, subject to some complications that will be explored later (see §3.10). My analysis is simply to make reference to an inviolable positional faithfulness constraint IdentIO([+stress])/_C_0# (202) that promotes faithfulness

to phonemic stress in the context of the final syllable of a stem.

(202) IdentIOFinal([+stress], C):

Short form — IdentIO([+stress])/_C_0#. *Output a violation for any word whose final vowel is unstressed if that vowel carried stress in the underlying form.*

The rankings used here are (203) and (204).

(203) IdentIO([+stress])/_C_0# $\gg_s$ NonFinality

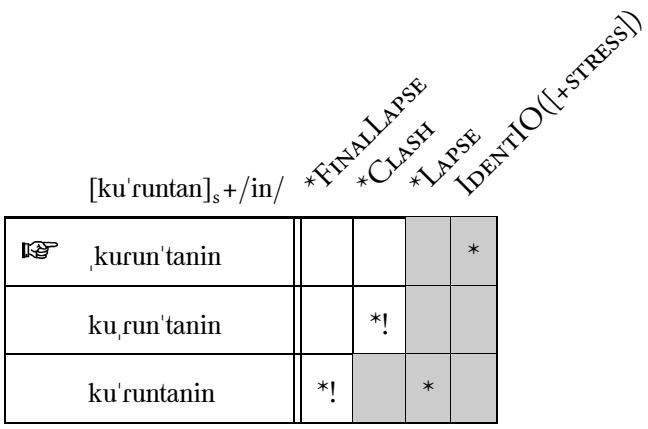
(204) *Lapse, *Clash $\gg_s$ IdentIO([+stress])

The ranking in (203) are illustrated in (205), and the rankings of (204) are partially illustrated in (206).

(205) /lehi'be/→[ˌlehi'be]$_s$ 'big'

/lehi'be/	IdentIO([+stress])/_C_0#	NonFinality
☞ ˌlehi'be		*
le'hibe	*!	

(206) [ku'runtan]$_s$+/in/ → [ˌkurun'tanin]$_s$ 'overthrow+*pass.*'



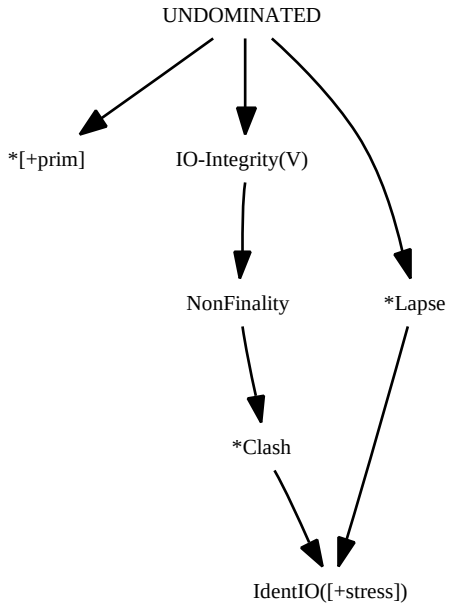
I cannot seem to unearth any examples of a morphologically simple four-syllable stem that submits to passivization, *e.g.* hypothetical [ˌala'helu]—[aˌlahe'luan], which is what would be necessary to show that *Lapse ≫ IdentIO([+stress]); at any rate, the tableau shows that either *FinalLapse or *Lapse must outrank IdentIO([+stress])).

3.8.4 RANKING SO FAR

3.8.4.1 *Stem Level*

Other than the generally undominated constraints (see §3.5.5), the following constraints have been established as undominated at the stem level: IdentIO([Cplace])/_V, *[αCplace] [−αCplace], WSP, *FinalLapse, IdentIO([+stress])/_C_0#.

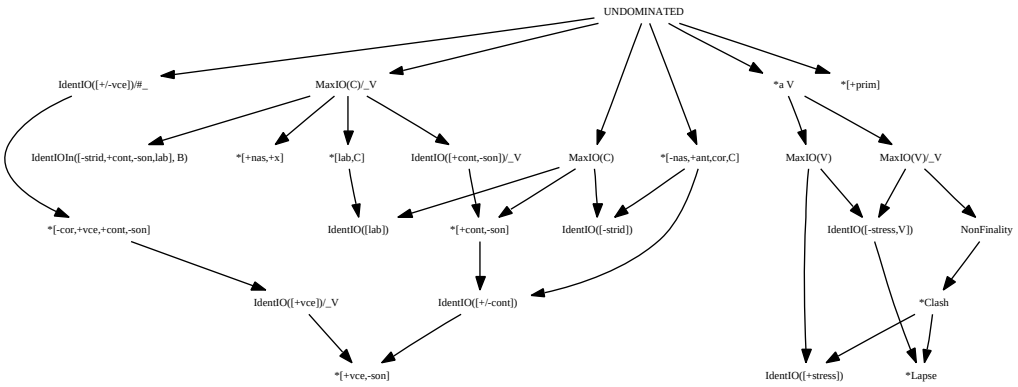
Stem-level rankings are as follows:



3.8.4.2 *Word Level*

Other than the generally undominated constraints, the following constraints have been established as undominated at the word level: IDENTIO([LAB])/_V, IDENTIO([±SYL]), *V_iV_i, *ea, *ei, and WSP.

The rankings of violable constraints are as follows, up to this point:



3.8.4.3 *Post-Lexical Level*

Other than adding WSP as undominated, the post-lexical level has been unaffected by this section.

3.9 CONSONANT-DELETING STEMS

Section 3.5 introduced a class of Malagasy roots that appear to be underlyingly consonant-final. However, in this analysis these roots are hypothesized to end in a limited set of consonants: /f/, /ɸ/, /m/, /t/, /ʧ/, /n/, /k/, and /h/. The analysis hypothesizes that roots that end in /b/, /p/, /d/, and /g/ would behave similarly, although such roots are unattested. There appears, however, to be another set of underlyingly consonant-final roots. The table in (207) shows a set of present indicative active forms paired with their circumstantial counterpart (except for (i), which has the present indicative active paired with the imperative).

(207)

Gloss	Isolation Form	Suffixed
(a) *to do*	ma'nâu̯	a'nâu̯van
(b) *to sun-dry*	mi'hahi	iha'hazan
(c) *to wait, watch*	mi'and͡ri	ian'd͡rasan
(d) *to take*	man'd͡râi	an'd͡râisan
(e) *to seek*	mi'tadi	itadi'avan
(f) *to kill*	maha'fati	ahafa'tesan
(g) *to go*	man'deha	ande'hanan
(h) *to buy*	mi'vidi	ividi'anan
(i) *to flatter*	man'duka	mandu'kafa

The present indicative active form is unsuffixed, the circumstantial form carries the suffix *-an*, and the imperative form carries the suffix *-a*. Note, however, that in these forms an unpredictable consonant is present in the suffixed form, but absent in the isolation form. There is also some vowel change going on; it will be analyzed in §3.10. The consonants that appear in forms like these are [v], [f], [z], [s], and [n]. The simplest explanation for these facts would be to say that the consonant that emerges is in fact underlying, and is deleted in the isolation form due to being ill-formed in some way. This is, in fact, essentially the explanation that I will give, but the simple form of it does not quite work. Recall that in roots ending with /f/, /ɸ/, /m/, /t/, /ʧ/, /n/, /k/, or /h/, the final consonant neutralizes in isolation, but is not deleted, and notice that this list of consonant endings includes two of the consonants that appear in the suffixed forms of (207) (that is, [f] and [n]). I should note that case (i) is in fact the only instance I have found of surface [f]

disappearing in the isolation form. In this case I have hypothesized that underlyingly this consonant is in fact /β/, a voiced non-strident bilabial fricative. The consonant /β/, it is hypothesized, deletes word-finally, hardens to [b] after a consonant (see §3.7.2.2), and surfaces as [f] elsewhere. As for surface [n], it is notable that [ŋ] appears in Malagasy words only preceding a velar consonant. My hypothesis here, then, is that velar nasals are marked in Malagasy, and therefore /ŋ/ appears on the surface as [n] when followed by a vowel, and does not appear at all word-finally. Thus, the proposed underlying consonants are as in (208).

(208)

Surface Consonant	Proposed UR
[v]	/v/
[f]	/β/
[z]	/z/
[s]	/s/
[n]	/ŋ/

First the analysis of deleting oral consonants, and then nasals.

3.9.1 DELETING STEM-FINAL ORAL CONSONANTS

3.9.1.1 *Deleting Alveolars*

Unlike alveolar stops and sonorants, alveolar fricatives delete at the end of a word or at the end of the base morpheme (except when the reduplicant begins with a consonant, as will be seen later). Tableau (211) shows that alveolar fricative preservation (209) prevents alveolar fricatives from changing into non-alveolars, and thus the only way to satisfy the general

(cross-linguistically unusual, see §3.7.3.2) dispreference for oral alveolars is to delete them, as shown in (211).

(209) IDENTIO([+CONT,-SON,+ANT,COR]):

Output a violation for each instance of an underlying alveolar fricative that corresponds to a surface segment that is not an alveolar fricative.

(210) IDENTIO([+CONT,-SON,+ANT,COR]), *[-NAS,+ANT,COR,C] $\gg_w$ MAXIO(C)

(211) /m+aha+fates/ → /m/+[ˌahaˈfates]$_s$ → [ˌmahaˈfati]$_w$ '*pres.*+*act.*+kill'[32]

/m/+[ˌahaˈfates]$_s$	IDENTIO([+CONT,-SON,-ANT,COR])	*[-NAS,+ANT,COR,C]	MAXIO(C)	*[+CONT,-SON]	IDENTIO([±CONT])
☞ ˌmahaˈfatiØ		*	*	**	
ˌmahaˈfatis		**!		***	
ˌmahaˈfatiʃ	*!	*		***	*

This deletion does not occur prevocalically due to the ranking

(212) MAXIO(C)/_V $\gg_w$ *[-NAS,+ANT,COR,C],

as shown in the following tableau:

[32]The vowel raising in this form will be discussed in §3.10.

(213) /aha+fates+an/ → [ˌahaˈfates]$_s$+/an/ → [ˌahafaˈtesan]$_w$ '*act.*+kill+*circum.*'

[aˌhafaˈtesan]$_s$	MaxIO(C)/_V	*[-nas,+ant,cor,C]	MaxIO(C)	*[+cont,-son]
☞ ˌahafaˈtesan		**		***
ˌahafaˈteØan	*!	*	*	**

Finally, as can be seen in tableaux (214) and (215), the same rankings that account for /s/ account for /z/ as well.

(214) /m+an$_x$+dihiz/ → /m/+[an$_x$ˈdihiz]$_s$ → [man$_x$ˈdihi]$_w$ '*pres.*+*act.*+dance'

/m/+[an$_x$ˈdihiz]$_s$	IdentIO([+cont,-son,-ant,cor])	IdentIOIn([+vce,+cont,-son], [+strid])	*[-nas,+ant,cor,C]	MaxIO(C)	*[+cont,-son]	IdentIO([±cont])	*[+vce,-son]
☞ man$_x$ˈdihiØ			*	*	*		*
man$_x$ˈdihiz			**!		**		**
man$_x$ˈdihiʧ	*!	*	*		**	*	*

(215) /an$_x$+dihiz+an/ → [an$_x$ˈdihiz]$_s$+/an/ → [an$_x$diˈhizan]$_w$ '*act.*+dance+*circum.*'

[an$_x$ˈdihiz]$_s$+/an/	MaxIO(C)/ _ V	*[-nas,+ant,cor,C]	MaxIO(C)	*[+cont,-son]	*[+vce,-son]
☞ an$_x$diˈhizan		**		**	**
an$_x$diˈhiØan	*!	*	*	*	*

3.9.1.2 *Deleting Labials*

Section 3.7.3 illustrates two alternations involving stem-final labials: [ʧ]:f in, *e.g.,* [manˈd͡ra-kuʧa]:[and͡raˈkufan] 'to cover'; and [k]:[f] in, *e.g.,* [miˈlelaka]:[ileˈlafan] 'to lick.' These were explained as reflexes of stem-final underlying /f/ and /ɸ/, respectively. There are two other similar alternations to be explained: Ø:[v] in, *e.g.,* [miˈtadi]:[i¸tadiˈavan] 'to seek,' and Ø:[f] in [manˈduka]:[anduˈkafan] 'to flatter.' I account for these as underlying stem-final /v/ and /β/, respectively. The feature-altering forms are underlyingly voiceless, and the deleting forms are underlyingly voiced.

Underlying /β/ Recall from §3.7.3.2 that the ranking in (216) causes /f/ and /ɸ/ to resist deletion and change place from labial to dorsal.

(216) MaxIO(C), *[lab,C] ≫$_w$ IdentIO([±cont]), IdentIO([lab])

This does not happen with the underlyingly voiced fricatives. First of all, /β/ does not change to [k] as /ɸ/ does because of IdentIOIn([-stri,+cont,-son,lab],β) (133), repeated here as (217).

(217) IdentIOIn([-stri,+cont,-son,lab],β):

Output a violation for each instance of /β/ that changes into something other than a non-strident labial fricative; i.e., */β/ can delete, remain the same, or change into [ɸ], but nothing else.*

(218) /dukaβ/ → [ˈdukaβ]$_s$ → [ˈduka]$_w$ 'flatter'

[ˈdukaβ]$_s$	IdentIOIn([-stri,+fric,lab],β)	*[lab,C]	MaxIO(C)	*[-cor,+vce,+fric]	*[+fric]	IdentIO([±cont])	IdentIO([-strid])	*[+vce,-son]
☞ ˈduka-			*					*
ˈdukaβ		*!		*	*			**
ˈdukaɸ		*!			*			*
ˈdukav	*!	*		*	*		*	**
ˈdukak	*!					*		*

(219) /m+an$_{\chi}$+dukaβ+a/ → /m/+[an$_{\chi}$ˈdukaβ]$_{s}$+/a/ → [man$_{\chi}$duˈkaɸa]$_{w}$ '*pres.*+*act.*+flatter+*imp.*'

[man$_{\chi}$ˈdukaβ]$_{s}$+/a/	IdentIOIn([-stri,+fric,lab], β)	*[-cor,+vce,+fric]	IdentIO([-strid])	IdentIO([+vce])/o[_V]	*[+vce,-son]
☞ man$_{\chi}$duˈkaɸa				*	*
man$_{\chi}$duˈkaβa		*!			**
man$_{\chi}$duˈkava	*!	*	*		**

(220) IdentIOIn([-stri,+cont,-son,lab], β), *[lab,C] $\gg_{w}$ MaxIO(C)

The ranking shown above in (220) causes the consonant to delete word-finally. In the suffixed form, the ban on peripheral voiced fricatives *[-cor,+vce,+cont,-son] (221) outranking IdentIO([+vce])/_V (222), ranking in (223), causes /β/ to become voiceless. The final step from [ɸ] to [f] comes at the postlexical level, due to an undominated ban on labial nonstrident fricatives (224), shown in tableau (225).

(221) *([-cor,+vce,+cont,-son]):

Output a violation for any instance of a surface peripheral (non-coronal) voiced fricative.

(222) IdentIOBefore([+vce], V):

Output a violation for any instance of an underlyingly voiced prevocalic segment with a voiceless surface correspondent. Short form—IdentIO([+vce]) / _ V.

(223) *[-cor,+vce,+cont,-son] $\gg_w$ IdentIO([+vce])/_V.

(224) *([-strid,+cont,-son,lab]):

Output a violation for each instance of a surface non-strident labial fricative.

(225) /m+an$_x$+dukaβ+a/ → /m/+[an$_x$'dukaβ]$_s$+/a/ → [man$_x$du'kaɸa]$_w$ → [mandu'kafa]

'*pres.*+*act.*+flatter+*imp*.'

[man$_x$du'kaɸa]$_w$	*[-strid,+cont,-son,lab]
☞ mandu'kafa	
mandu'kaɸa	*!

The change from /β/ to [ɸ]$_w$ and thence to [f] does not occur word-initially, as seen earlier in §3.7.2.2. This is because a constraint preserving voicing word-initially (226) outranks the ban on peripheral voiced fricatives (221), as shown in tableau (228).

(226) IdentIOInitial([±vce]):

Output a violation for any instance where an underlyingly word-initial segment corresponds to a surface segment that differs from it in the feature [±vce]. Short form—IdentIO([±vce])/#_.

(227) IdentIO([±vce])/#_ $\gg_w$ *[-cor,+vce,+cont,-son].

(228) /βule+an/ → [βu'lean]$_s$ → [βu'len]$_w$ 'plant+*pass.*'

[βu'lean]$_s$	IdentIOIn([-strid,+cont,-son,lab], β)	IdentIO([+cont,-son])/_V	IdentIO([+/-vce])/#_	*[+cont,-son]	IdentIO([+/-cont])	*[-cor+vce,+cont,-son]	IdentIO([+vce])/_V	*[+vce,-son]
☞ βu'leØn				*		*		*
ɸu'leØn			*!	*			*	
bu'leØn	*!	*			*			*

Underlying /v/ Underlying /v/ does not become [ʧ] as /f/ does because the quality of being a voiced fricative is better preserved in strident fricatives, due to the following undominated constraint:

(229) IdentIOIn([+vce,+cont,-son],[+strid]):

Output a violation for every instance of an underlying strident segment that is underlyingly a voiced fricative but corresponds on the surface to something that is not a voiced fricative.

This is shown in tableaux (230) and (231).

(230) /laluv/ → [ˈlaluv]$_s$ → [ˈlalu]$_w$ ‘blind’

[ˈlaluv]$_s$	IdentIOIn([+vce,+cont,-son], [+strid])	*[Lab, C]	MaxIO(C)	*[+cont,-son]	IdentIO([±cont])	*[-cor,+vce,+cont,-son]	*[+vce,-son]	IdentIO([lab])
☞ ˈlaluØ			*					
ˈlaluv		*!		*		*	*	
ˈlaluʧ	*!			*	*			*

(231) /laluv+an/ → [laˈluvan]$_s$ → [laˈluvan]$_w$ ‘blind+*pass.*’

[laˈluvan]$_s$	MaxIO(C)/_V	*[Lab, C]	MaxIO(C)	*[+cont,-son]	*[-cor,+vce,+cont,-son]	*[+vce,-son]
☞ laˈluvan		*		*	*	*
laˈluØan	*!		*			

The first form shows that the ranking

(232) IdentIOIn([+vce,+cont,-son], [+strid]), *[lab,C] $\gg_w$ MaxIO(C)

leads to similar results as for /β/. The usual constraints IdentIO([lab])/_V, MaxIO(C)/ _V and IdentIO([+vce])/ _V are responsible for [v] in the suffixed form.

3.9.2 DELETING STEM-FINAL NASAL CONSONANTS

Underlying word-final /ŋ/ deletes because of a general ban on dorsal nasals (233) outranking MAXIO(C). The consonant deletes instead of changing because a constraint against changing place in dorsal nasals (234) outranks MAXIO(C) as well (235).

(233) *([+NAS,DORS])

Output a violation for each surface instance of a dorsal nasal segment.

(234) IDENTIO([+NAS,DORS])

Output a violation for any underlying dorsal nasal segment that corresponds to a surface segment which is not a dorsal nasal.

(235) *[+NAS,DORS], IDENTIO([+NAS,DORS]) $\gg_w$ MAXIO(C),

This is shown in tableau (236).

(236) /m+an$_x$+lehaŋ/ → /m/+[an$_x$ˈlehaŋ]$_s$ → [man$_x$ˈleha]$_w$ '*pres.*+*act.*+go'

/m/+[an$_x$ˈlehaŋ]$_s$	*[+NAS,DORS]	IDENTIO([+NAS,DORS])	MAXIO(C)
☞ man$_x$ˈlehaØ			*
man$_x$ˈlehan		*!	
man$_x$ˈlehaŋ	*!		

In prevocalic position the /ŋ/ appears as [n] because the ban on dorsal nasals (233) and the ban on prevocalic consonant deletion (MaxIO(C)/_V (78)) outrank the relevant IdentIO constraint as shown in (237), so feature change results. See tableau (238) for an illustration of this.

(237) *[+nas,dors], MaxIO(C)/_V $\gg_w$ IdentIO([+nas,dors]).

(238) /an$_x$+lehaŋ+an/ → [an$_x$'lehaŋ]$_s$+/an/ → [an$_x$le'hanan]$_w$ '*act.*+go+*circum.*'

[an$_x$'lehaŋ]$_s$+/an/	*[+nas,dors]	MaxIO(C)/_V	IdentIO([+nas,dors])	MaxIO(C)	NonFinality	MaxIO(V)
☞ an$_x$le'hanan			*			
an$_x$le'haØØn		*!		*	*	*
an$_x$le'haŋan	*!					

3.10 VOWEL WEAKENING

As has already been demonstrated, Malagasy has a number of consonant-neutralizing phenomena at the right word boundary. There appear to be two word-final vowel reduction phenomena as well, both having to do with front non-high vowels.

3.10.1 MID TO HIGH WEAKENING

The following forms illustrate a frequent pattern of [e]–[i] alternation in Malagasy.

(239)

Gloss	Unsuffixed	Suffixed
to kill	ˌmahaˈfati	aˌhafaˈtesan
to please	maˌhafiˈnaritʃa	ˌahaˌfinaˈretan
to sit	miˈpetʃaka	mipeˈʧaha
to speak a dialect	miˈruki	ˌiruˈkian
to look at	miˈʤeri	ˌiʤeˈren

There are some forms where [i] alternates with [e], some where [e] appears in both forms, and some where [i] appears consistently. This would seem to be a completely random affair, then, except that there are some consistent features of the data. First, in the forms that do alternate, the [i] alternate appears in the immediate posttonic syllable. In all of the forms, [e] appears in or before the main stress of the word; [i] may appear anywhere.

Due to these characteristics, I have chosen to analyze the situation as one where the underlying vowel of the alternation is /e/. The alternation is accounted for at the word level partially because at that level the change from /e/ to [i] occurs in an easily defined location—the last syllable of the word, if that syllable does not have phonemic stress. The explanation for the phenomenon is grounded primarily in an effort minimization constraint against high-sonority front vowels at the right edge of the word:

(240) *Final([-back,-high], C):

Short form—*[-back,-high] C_0#. *Output a violation for any word in which the last vowel is front and not high.*

Heightened vowel faithfulness in the environment of underlying stress (241) or prevocalic position (242) counteracts the effort minimization constraint—ranking (243).

(241) IDENTIOIN([-HIGH], [+STRESS]):

Output a violation for each underlying stressed non-high vowel that has a surface correspondent that is high.

(242) IDENTIOBEFORE([-HIGH], V):

Short form—IDENTIO([-HIGH])/_V. *Output a violation for each underlyingly prevocalic segment that is non-high but has a high surface correspondent.*

(243) IDENTIO([-HIGH])/_V, MAXIO([+STRESS]), IDENTIOIN([-HIGH], [+STRESS]), MAXIO(V)/_V $\gg_w$ *[-BACK,-HIGH] C_0# $\gg_w$ IDENTIO([-HIGH])

The constraint IDENTIO([-HIGH])/_V essentially takes care of situations where a word-final environment is created by vowel deletion[33].

The ranking is demonstrated in tableaux (244)–(246).

[33] This could be a situation where an argument for surface contextual faithfulness (in IDENTIOIN([-HIGH], [+STRESS])) might legitimately be made. That is, if the [+STRESS] context were taken to be a surface context rather than underlying, it might not be necessary to employ IDENTIO([-HIGH])/_V to rule out candidates like *[midʑeˈria] of tableau (246). I have not investigated this possibility in detail due to the amount of time it would take to reconfigure the system for this, but I imagine that there would be some problems with the fact that IDENTIOIN([-HIGH], SURFACE([+STRESS])) would act as an implicit *FINAL([+STRESS,-HIGH], C) constraint.

(244) /m+an$_x$+etʃe/ → /m/+[an$_x$ˈetʃe]$_s$ → [man$_x$ˈetʃi]$_w$ ‘*pres.*+*act.*+demote’

/m/+[aˈn$_x$etʃe]$_s$	*[-back,-high] C$_0$#	IdentIO([-high])
☞ maˈn$_x$etʃi		*
maˈn$_x$etʃe	*!	

(245) /ˈbe/ → [ˈbe]$_s$ → [ˈbe]$_w$ ‘numerous’

[ˈbe]$_s$	IdentIOIn([-high], [+stress])	*[-back,-high] C$_0$#	IdentIO([-high])
☞ ˈbe		*	
ˈbi	*!		*

(246) /m+i+dzere+a/ → /m/+[iˈdzere]$_s$+/a/ → [midzeˈre]$_w$ ‘*pres.*+*act.*+look at+*imp.*’

/m/+[idzere]$_s$+/a/	IdentIO([-high])/_V	MaxIO(V)/_V	*[-back,-high] C$_0$#	NonFinality	MaxIO(V)	IdentIO([-stress, V])	IdentIO([-high])	*Lapse	IdentIO([+stress])
☞ midzeˈreØ			*	*	*	*		*	*
miˈdzerØa		*!			*				
midzeˈria	*!					*	*	*	*

3.10.2 LOW TO HIGH WEAKENING

Malagasy has another, similar, alternation, but it is much less prevalent. In some of the forms given below, [a] and sometimes [ia] alternates with [i].

(247)

	Gloss	Unsuffixed	Suffixed
(a)	*to sun-dry*	mi'hahi	iha'hazan
(b)	*to wait, watch*	mi'and͡ri	ian'd͡rasan
(c)	*to seek*	mi'tadi	itadi'avan
(d)	*to buy*	mi'vidi	ividi'anan
(e)	*love each other*	mi˛faŋka'tia	fi˛faŋkati'avan
(f)	*be flattered*	man'duka	mandu'kafan
(g)	*fruit*	'vua	famu'azan

As can be seen from the last two forms in the table, this alternation does not always occur, and its occurrence or non-occurrence does not appear to be predictable from any quality of the phonological context. The phenomenon may be summarized as follows: in forms where a stem consonant appears in a combined form but not in isolation (these forms being the ones discussed in §3.9) and where the vowel that appears just before the deleting stem consonant is [a], the isolation form has the vowel [i] instead.

My analysis for these forms is that at the lexical level they are quite similar to the [e]–[i] alternations discussed above. I propose to set up the underlying form of the [a] vowels that rise to [i] as abstract /æ/. Looking back to the previous section, the same motive for raising /e/ to [i] applies in even greater force here, since [æ] is a higher-effort vowel, although this

is perhaps counterbalanced by the increased severity of the faithfulness violation. There are three basic patterns in the table above:

1. Like (a) and (b), where [a] alternates with [i].
2. Like (c) and (d), where [ia] alternates with [i].
3. Like (e)–(g), where no alternation occurs.

I will illustrate each of these cases with tableaux.

ALTERNATION OF [a] WITH [i] The behavior in cases (a) and (b) is due to the following ranking,

(248) *[-BACK,-HIGH] C_0#, MAXIO(V) $\gg_w$ IDENTIO([-HIGH]), IDENTIO([+LOW])

as shown in the tableau of (249).

(249) /m+i+hahæz/ → /m/+[iˈhahæz]$_s$ → [miˈhahi]$_w$ '*pres.*+*act.*+sun-dry'

/m/+[iˈhahæz]$_s$	*[-BACK,-HIGH] C_0#	MAXIO(V)	IDENTIO([-STRESS,V])	IDENTIO([-HIGH])	IDENTIO([+LOW])	IDENTIO([+STRESS])
☞ miˈhahiØ				*	*	
ˈmihahØØ		*!	*			*
miˈhahæØ	*!					

In the suffixed form there is no pressure to reduce the vowel, and it is maintained as is due to faithfulness constraints, as shown in tableau (250).

(250) /i+hahæz+an/ → [iˈhahæz]$_s$+/an/ → [ihaˈhæzan]$_w$ '*act.*+sun-dry+*circum.*'

[iˈhahæz]$_s$+/an/	IDENTIO([-HIGH])	IDENTIO([+LOW])
☞ ihaˈhæzan		
ihaˈhezan		*!
ihaˈhizan	*!	*

This vowel then changes to its [+BACK] equivalent [a] at the postlexical level through a ban on low front vowels:

(251) /i+hahæz+an/ → [iˈhahæz]$_s$+/an/ → [ihaˈhæzan]$_w$ → [ihaˈhazan] '*act.*+sun-dry+*circum.*'

[ihaˈhæzan]$_w$	*[-BACK,+LOW]	IDENTIO([±LOW])	*[-LOW]
☞ ihaˈhazan			*
ihaˈhezan		*!	**
ihaˈhæzan	*!		*

ALTERNATION OF [ia] WITH [i] These cases, like the first, have *[-BACK,-HIGH] C_0# eliminating [æ] from the right edge of a form, but here it is by deletion rather than by raising. A vowel raising solution to the problem of a word-final non-high front vowel is ruled out because it would create an illegal hiatus. Normally, as discussed in §3.8.1, hiatus avoidance creates an opaque stress, but here it does not, because deleted front vowels do not trigger compensatory lengthening via the OPQSTRESS constraint (perhaps they are less perceptually salient). Because of its specificity for deleted back vowels, OPQSTRESS does not play a part in the analysis of these (second case) forms, so the relevant ranking is simply that shown in (252).

(252) *i i, *[-BACK,-HIGH] C_0#, NONFINALITY $\gg_w$ MAXIO(V), IDENTIO([-STRESS,V]), IDENTIO([+STRESS])

This is illustrated in tableau (253).

(253) /m+i+tadiæv/ → /m/+[ˌitaˈdiæv]$_s$ → [miˈtadi]$_w$ '*pres.*+*act.*+look-for'

/m/+[ˌitaˈdiæv]$_s$	*ɪ ɪ	*[-BACK,-HIGH] C_0#	NONFINALITY	MAXIO(V)	IDENTIO([-STRESS,V])	IDENTIO([-HIGH])	IDENTIO([+LOW])	*LAPSE	IDENTIO([+STRESS])
☞ miˈtadiØØ				*	*				**
ˌmitaˈdiØØ			*!	*					
ˌmitadiˈe͡ɪØ			*!		*	*	*	*	*
ˌmitadiˈæ͡ɪØ			*!		*	*	*	*	*
ˌmitaˈdiæØ		*!							
ˌmitaˈdiiØ	*!					*	*		

NON-ALTERNATION Forms (e)–(g) do not alternate, and need little explanation. When the underlying form is /a/, the constraint *[-BACK,-HIGH] C_0# no longer applies and therefore no change is made. This is illustrated in (254).

(254) /vuaz/ → ['vuaz]$_s$ → ['vua]$_w$ 'fruit'

['vuaz]$_s$	IDENTIO([±BACK])	*[-BACK,-HIGH] C_0#	IDENTIO([-HIGH])	IDENTIO([+LOW])
☞ 'vuaØ				
'vueØ	*!	*		*
'vuiØ	*!		*	*

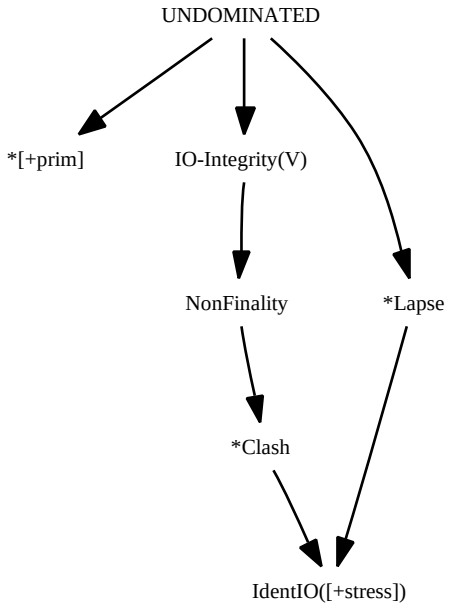
3.10.3 ANALYSIS SO FAR

Before moving into an analysis of reduplication, it will be worthwhile to review the current constraint rankings and the phenomena that occur at each level of analysis.

3.10.3.1 *Stem Level*

The following constraints have been established so far as undominated at the stem level: *|[+SON] [−SON]|, *|[α C-PLACE] [−α C-PLACE]|, *|[−CONT] [+CONT, −COR]|, *|[α NAS] [−α NAS]|, *|[+CONT] [−CONT]|, *|[α VCE] [−α VCE]|, *t̲ X, *t̲#, *X ʃ, *#ʃ, *|ei|, *|[+HIGH] [−HIGH]|, *|[−LOW] [+LOW]|, *|CV|, *|VC|, IDENTIO([CPLACE]) / _ V, *[α C-PLACE] [−α C-PLACE], WSP, *FINALLAPSE, IDENTIO([+STRESS]) / _ C_0#.

The following diagram shows the current relative rankings of the stem level violable constraints:



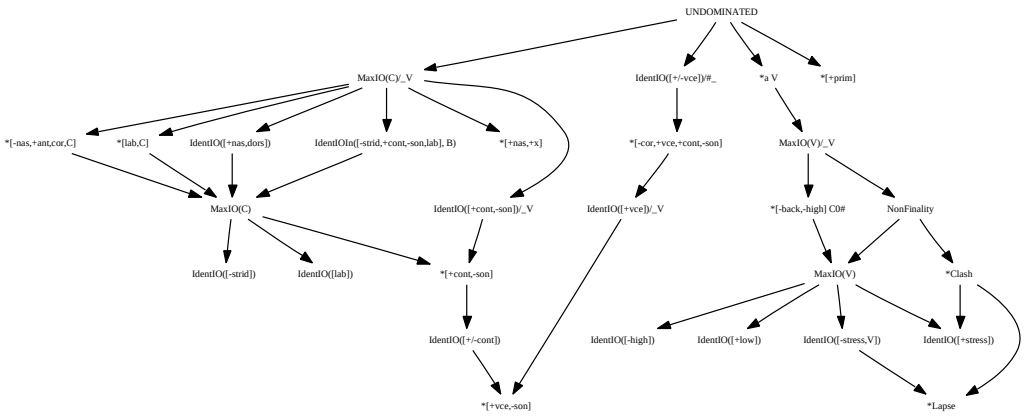
The changes induced by this level on an input form are as follows:

- Stress is assigned in right-to-left trochees, with the exception that stress always falls on a diphthong, and stress may be lexically marked as word-final.
- The rightmost stress of a word is marked as primary, the rest as secondary.
- From right to left, a pre-consonantal consonant assimilates in place to the following consonant.

3.10.3.2 *Word Level*

The following constraints have been established so far as undominated at the word level: *|[+SON] [−SON]|, *|[αCPLACE] [−αCPLACE]|, *|[−CONT] [+CONT, −COR]|, *|[αNAS] [−αNAS]|, *|[+CONT] [−CONT]|, *|[αVCE] [−αVCE]|, *t̲ X, *t̲#, *X ʃ, *#ʃ, *|ei|, *|[+HIGH] [−HIGH]|, *|[−LOW] [+LOW]|, *|CV|, *|VC|, IDENTIO([LAB]) / _ V, IDENTIO([±SYL]), *V_i V_i, *ea, *ei, WSP, and IDENTIO([+CONT, −SON, +ANT, COR]), IDENTIO([−HIGH]) / _ V, and IDENTIOIN([-HIGH], [+STRESS]).

The following diagram shows the current relative rankings of the word level violable constraints:



The changes induced by this level on an input form are as follows:

- If a word-level suffix is added, the final stress in the word shifts to the right to ensure primary stress on the penultimate syllable.

- Disallowed vowel sequences (V_iV_i, a*V*, ea, and ei) are resolved in favor of the vowel that bore stress in the input. If neither bore stress, the leftmost vowel survives. In

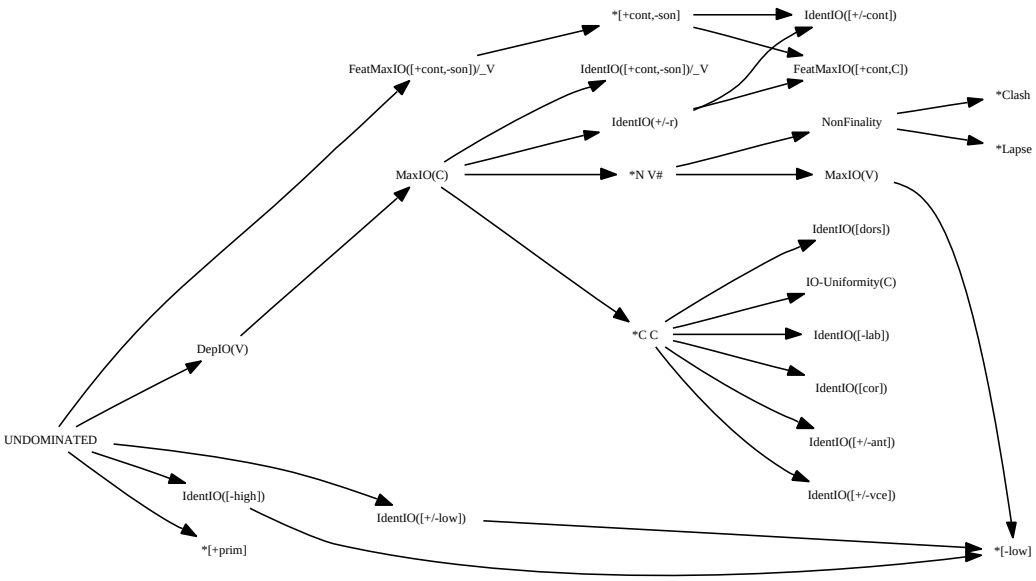
this latter case, the leftmost vowel acquires stress if the rightmost vowel was [+BACK].

- Word-final consonants either neutralize to one of {[n], [ʧ], [k]}, or they delete.
- Word-final non-high front vowels raise to [i].
- Voiceless consonants and voiced strident labial fricatives delete after the prefix *-an*.

3.10.3.3 *Post-Lexical Level*

The following constraints have been established so far as undominated at the post-lexical level: *|[+SON] [−SON]|, *|[αCPLACE] [−αCPLACE]|, *|[−CONT] [+CONT, −COR]|, *|[αNAS] [−αNAS]|, *|[+CONT] [−CONT]|, *|[αVCE] [−αVCE]|, *t̲ X, *t̲#, *X ʃ, *#ʃ, *|ei|, *|[+HIGH] [−HIGH]|, *|[−LOW] [+LOW]|, *|C V|, *|V C|, WSP, DEPIO(C), MAXIO([+STRESS]), MAXIO([−NAS, C]), IDENTIO([±NAS]), *C [+NAS], IDENTIO([±SYL]), *[αCPLACE] [−αCPLACE], *[−CONT] [+CONT, C], and PTIDENTIO([−SON, −CONT]).

The following diagram shows the current relative rankings of the post-lexical level violable constraints:



The changes induced by this level on an input form are as follows:

- As in the other levels, the final stress of the word is marked as primary, others as secondary.
- Consonant clusters are simplified in various ways depending on their constituent segments and their position in a word.
- Unstressed vowels after a nasal at the end of the word are deleted.
- If the word ends in an oral consonant, the vowel [a] is introduced word-finally.

3.11 REDUPLICATION AND COMPOUNDING

Earlier sections have established the stem, word, and post-lexical levels and the morphological processes that occur at the stem and word levels. This leaves only post-lexical morphology, which I claim includes, in Malagasy, both genitival compounding and reduplication (with some exceptions in the case of reduplication). The phonological phenomena that occur at a compound boundary has already been covered in §§3.5 and 3.7, so I will discuss genitival compounding only to summarize how the morphology works in this analysis and then I will move on to a more complete coverage of reduplication.

3.11.1 GENITIVAL COMPOUNDING

As stated earlier, the genitive construction in Malagasy is a compound consisting, in order from left to right, of the object possessed, a nasal element, and finally the possessor. The procedure for creating a compound in this analysis is as follows:

1. Introduce the object possessed at the stem level, generating cyclically as usual to add stem-level affixes.

2. Introduce the possessor at the stem level, generating cyclically as usual, in the same manner as for the object possessed.

3. Take the output of step (1), add any word-level affixes, then use the word-level grammar to generate a word-level output.

4. Take the output of step (2) plus any word-level affixes, with the genitive prefix /n-/

attached at the beginning of the word[34], and generate a word-level output.

5. Concatenate the outputs of steps (3) and (4). This is the input to the post-lexical grammar, the output of which is the compound.

In summary, the parts of the compound go through the stem and word levels in isolation from one another, the genitive suffix being added to the first part of the compound at the word level, and the two parts are joined before going through the post-lexical level. The key here is that word-boundary effects and vowel hiatus resolution apply individually to the separate parts of the compound before it is joined together, so any vowel hiatus created by compounding remains unresolved, and word-boundary effects apply to both members of a compound.

3.11.1.1 *Example*

The above is best understood by an example. The word [ˌtuŋgutʃaˈkuhu] signifies "chicken's foot," and is composed of the morphemes /tuŋgut/ 'foot,' /n/ 'genitive,' and /akuhu/ 'chicken.' The derivation is as follows, according to the steps laid out above:

1. Stem (possessed object): /tuŋgut/ ⇒ [ˈtuŋgut]$_s$.

2. Stem (possessor): /akuhu/ ⇒ [aˈkuhu]$_s$.

3. Word (possessed object): [ˈtuŋgut]$_s$ ⇒ [ˈtuŋgutʃ]$_w$ (note application of word-final neutralization).

[34] This assumes that /n-/ is a word-level prefix. As far as the phonology of this analysis goes, it could just as well be a suffix on the possessed word or a separate word altogether.

4. Word (possessor with genitive prefix): /n/+[aˈkuhu]$_s$ ⇒[naˈkuhu]$_w$.

5. Post-lexical (entire compound): [ˈtuŋgutʃ]$_w$+[naˈkuhu]$_w$ ⇒[ˌtuŋgutʃaˈkuhu] (note simplification of [tʃn]$_w$ to [tʃ], as discussed in §3.7.3.3).

Actually, this derivation contains more information than the actual computational mechanism gets, because of LPM bracket erasure. The actual inputs at the five generations are unbracketed strings where stress is marked only on the stressed vowel itself (I will use acute accents here for primary stress [+STRESS,+PRIM] and grave for secondary [+STRESS,-PRIM]):

1. tuŋgut ⇒ túŋgut

2. akuhu ⇒ akúhu

3. túŋgut ⇒ túŋgutʃ

4. nakúhu ⇒ nakúhu

5. túŋgutʃnakúhu ⇒ tùŋgutʃakúhu

3.11.2 REDUPLICATION DATA

Here are the general patterns of Malagasy reduplicated forms, presented in an order that facilitates gradual discovery of what shape an analysis must take. Throughout this section I will attempt to decide between two basic hypotheses about how reduplication might occur:

1. **Correspondence Theory:** at some stage of the derivation, whether it be stem, word, or post-lexical, the morpheme "RED" is affixed. This morpheme is taken as having

as its underlying form the entirety of its base of affixation. For example, if the morpheme is affixed at the beginning of the stem level, its underlying form is the underlying form of the base. If affixed later at a later cycle, its underlying form is the output of the previous cycle. If affixed at the word level, its underlying form is the output of the stem level, and so forth. During this level, and during this level alone, since LPM-OT bracket erasure will eradicate evidence of a reduplicant and a base before the next level is entered, base-reduplicant correspondence will play a role in ensuring that the base and reduplicant are as similar as possible. At this level and at subsequent levels, input-output correspondence will also play a role. The key features of the correspondence theory approach are that reduplication is treated as affixation and that there are constraints requiring that the surface form of the base be similar to the surface form of the reduplicant.

2. **Morpheme Doubling:** A reduplicating form enters the derivation process as two separate underlying forms (essentially, two separate words). Both of the forms contain an instance of the same root morpheme, but they may also contain affixes. For example, the form "mi¡salasa'la" (*'pres.+act.*+hesitate+*red.+imp.'*) would begin its derivation process as two forms: /m+i+sala/ and /sala+a/ where the root morpheme of the first form is marked as being a base and the second is marked as being a reduplicant. At some stage in the derivation the two forms merge, as in compounding, and proceed thenceforth as a single form. The key features of the morpheme doubling approach are that reduplication is treated as compounding and that there are no constraints requiring that the surface form of the base be similar to the surface form of the reduplicant. The morpheme doubling framework here

essentially follows Inkelas & Zoll (2000).

Once one of these frameworks is decided upon, it will then become necessary to decide, in the case of Correspondence Theory, at which stage the reduplicant is affixed, or, in the case of Morpheme Doubling, at which stage the independent forms become merged.

3.11.2.1 *Two-Syllable Vowel-Final Paroxytones*

(255) below shows the reduplication pattern for simple stems consisting of a single trochaic foot.

(255)

Gloss	Simple	Reduplicated
(a) *white*	ˈfutsi	ˌfutsiˈfutsi
(b) *stinky*	ˈmâ͡imbu	ˌmâ͡imbuˈmâ͡imbu
(c) *different*	ˈhafa	ˌhafaˈhafa

It is difficult to come to any firm conclusions from these simple reduplicated forms. It appears clear that reduplication involves copying a stem, but it is not yet clear whether the copy is placed to the left or right of the base stem. Nor is it clear whether the reduplication is total or partial. It is total here, but the bases are small. At any rate it is already apparent that the reduplicant may comprise more than one syllable.

3.11.2.2 *One-Syllable Vowel-Final Stems*

(256)

Gloss	Simple	Reduplicated
(a) *big, numerous*	ˈbe	ˌbeˈbe
(b) *rotten*	ˈlu	ˌluˈlu

One new fact becomes clear from these forms: stress class of a type unacceptable in unreduplicated forms can be found in reduplicated ones. There could be several explanations for this:

1. (Correspondence Theoretic) Base-Reduplicant stress identity (IDENTBR([±STRESS])) outranks the ban on stress clash.

2. (Either) Stress assignment precedes reduplication, *i.e.* reduplicant affixation or base-reduplicant form merger occurs after the word level, since *CLASH outranks IDENT-IO([+STRESS]) at the word and stem levels.

3. Stress rankings have base- or reduplicant-specific exceptions, for example, IDENT-IO([+STRESS]) / $_{\text{RED}}$[_] ≫*CLASH.

By Occam's Razor explanations (1) and (2) may be preferable to (3).

3.11.2.3 *Multi-Syllabic Vowel-Stem Paroxytones*

(257)

Gloss	Simple	Reduplicated
(a) *sadness*	ˌalaˈhelu	ˌalaˌhelu-ˈhelu

This example shows that reduplication is partial, not total. Further, there are only two plausible theories as to the placement of the reduplicant with respect to the base: either the reduplicant is suffixed, or it is infixed just before the final foot of the base. For the sake of simplicity I will assume that reduplication is suffixing, and it will become evident that this assumption works out reasonably well.

3.11.2.4 *Two-Syllable Vowel-Stem Oxytones*

(258)

Gloss	Simple	Reduplicated
(a) *again*	in'd͡rai̯	in,d͡rai̯-n'd͡rai̯
(b) *barking*	vu'vu	vu,vu-'vu

The bases in these examples have final stress due to WSP (a) or phonemic stress (b). Only the stressed syllable reduplicates. This implies that the reduplicant does not follow a two-syllable template, but instead is a left-headed foot.

3.11.2.5 *Underlying Nasal + Vowel Final Stems*

The unreduplicated forms in the following table end in a nasal, and the final syllable is stressed.

(259)

Gloss	Simple	Reduplicated
(a) *forget*	ha'din	ha,dinu-'din
(b) *healthy*	sa'lam	sa,lama-'lam

Forms like these are analyzed in §3.5.3 as ending underlyingly in a vowel which is deleted at the post-lexical level. The reduplicated form confirms this — an unpredictable vowel

appears between the base and the reduplicant. These forms narrow the choice of analysis somewhat—either reduplication must occur before the post-lexical level or if a correspondence-theoretic reduplication occurs at the post-lexical level, *Clash and *NV# must outrank MaxBR(V).

3.11.2.6 *Vowel-Initial Weak Stems*

(260)

Gloss	Simple	Reduplicated
(a) *bouncing back*	ˈevutʃa	ˌevutʃ-ˈevutʃa
(b) *baggage*	ˈentan	ˌentaˈn-entan
(c) *twist*	ˈulika	ˌuliˈk-ulika

In a one-level or post-lexical correspondence-theoretic account, these forms would force an analysis wherein the reduplicant is the two syllables after the primary stress, infixed to the left of the rightmost foot in the base. In a multi-level account, however, it suffices to say that reduplication occurs either at the word level (for a Correspondence Theoretic account) or between the word level and the post-lexical level (for a morpheme doubling account), so word-final consonant neutralization is reflected in both the base and the reduplicant, but the epenthetic vowel, which does not exist at the word level, is not copied.

3.11.2.7 *Consonant-Initial Weak Stems*

(261)

Gloss	Simple	Reduplicated
(a) *thing*	ˈzavatʃa	ˌzavaˈʤavatʃa
(b) *veranda*	ˌlavaˈraŋgan	ˌlavaˌraŋganˈd͡raŋgan
(c) *known*	ˈfantatʃa	ˌfantaˈpantatʃa
(c) *far*	ˈlavitʃa	ˌlaviˈdavitʃa

These forms show that weak roots, when reduplicated, introduce consonant clusters that are resolved as laid out in §3.7. This is further evidence that reduplication occurs before the post-lexical level.

3.11.2.8 *Reduplication with Suffixes*

The following forms are quite paradoxical.

(262)

Gloss	Simple	Passive	Reduplicated	Reduplicated Passive
(a) *offer*	ˈtulutʃa	tuˈluran	ˌtuluˈtulutʃa	ˌtulutuˈluran
(b) *known*	ˈfantatʃa	fanˈtarin	ˌfantaˈpantatʃa	ˌfantapanˈtarin
(c) *redemption*	ˈavutʃa	aˈvutan	ˌavuˈʧavutʃa	ˌavutʃaˈvutan

Earlier I have analyzed the final consonant in the simple forms above as being a neutralized version of the consonant revealed by the passive form. The consonant is neutralized when word-final. However, the neutralized form appears in the reduplication despite not being word-final. In the active (unsuffixed) reduplicated form, a Correspondence-Theoretic analysis might analyze the neutral consonant in the base as being a case where

Base-Reduplicant identity has caused the reduplicant to be reflected in the base, but in the suffixed form, the neutralized consonant appears in the base with no place for it to have been copied from! The expected form from a Correspondence Theoretic perspective would be, *e.g.* [ˌavutaˈvutan]. The morpheme doubling analysis works quite well here, however – if we take the base of reduplication as one independent form and the reduplicant plus the suffix as another and have them proceed independently through the stem and word levels and then join just prior to the post-lexical level, the data pattern falls out exactly as given above:

(263)

	Base	Reduplicant+Passive Sfx
UR	/avut/	/avut+an/
Stem	[ˈavut]$_s$	[aˈvutan]$_s$
Word	[ˈavutʃ]$_w$	[aˈvutan]$_w$
Post-Lexical	[ˈavutʃaˈvutan]	

Note that such forms are also highly problematic for a Correspondence Theoretic analysis in their stress pattern as well. If ranking identity of stress pattern above conformance to the basic stress pattern of the language is responsible for forms such as [ˌbeˈbe] and [vuˌvuˈvu], why do we get forms such as [ˌavutʃaˈvutan] where both the basic stress pattern *and* base-reduplicant identity are violated? After all, the form *[aˌvutaˈvutan] is better on both counts. A Correspondence Theoretic account would need to treat this pattern in terms of something like Paradigm Uniformity, *i.e.,* an output-output correspondence version of the analysis given here, leaving Base-Reduplicant identity with essentially no rôle in the grammar. The stress pattern seen here falls out straightforwardly from a morpheme

doubling account paired with LPM-OT.

3.11.2.9 *Early and Late Reduplication*

From the forms seen so far, it seems that a straightforward account of the reduplication pattern of Malagasy is possible—reduplicated forms are derived as in (263), where the base and any prefixes proceed as one unit through the stem and word levels, and the reduplicant plus any suffixes proceed as another unit through these levels, and then the two are merged just prior to the post-lexical level. As data set (264) shows, however, there are still a few complications.

(264)

Gloss	Simple	Passive	Reduplicated	Red. Pass.
(a) *coward*	ˈusa	—	ˌusaˈusa	—
(b) *redemption*	ˈavutʃa	aˈvutan	ˌavuˈtʃavutʃa	ˌavutʃaˈvutan
(c) *wander about*	ˈreni	ɾeˈnen	ˌreniˈren	—
(d) *change*	ˈuva	—	ˌuˈvuva	—
(e) *spit*	ˈivi	—	ˌivˈivi	—
(f) *ray of light*	ˈhiran	—	ˌhiraŋˈgiran	—
(g) *blind*	ˈlalu	laˈluvan	ˌaluˈdalu	ˌaludaˈluvan

Compare (a) with (d). The forms are nearly identical, but (d) exhibits vowel merger, whereas (a) does not. It should be noted that for (d) [ˌuˈvuva] is in fact in free variation with [ˌuvaˈuva]. As shown in §3.8.1, vowel merger occurs at the word level, so the prediction of the previous section would be that the data would pattern as (a), not (d). One account for this difference would be to hypothesize that in (d) reduplication occurs *before*

the word level rather than after it. Another hypothesis would take the reduplicated form of (d) as being a lexicalized form with no base-reduplicant marking. I believe the former hypothesis to be preferable on the grounds that these forms pattern semantically like the other reduplicated forms. The free variation might tend to support the former hypothesis as well. The forms in (a)–(c) all exhibit evidence of being *late reduplications*: (a) has been covered, (b) exhibits word-final consonant neutralization in what would otherwise have been a word-internal consonant, and (c) exhibits word-final vowel weakening in what would otherwise have been a word-internal position. Forms (d)–(g) are *early reduplications.* Forms (d) and (e) exhibit vowel merger, indicating hiatus at the word level. For form (f) I must hypothesize the underlying form /ɣiran/. In isolation, such an underlying form would become [ˈhiran]$_w$ before the post-lexical level, and therefore its predicted reduplicated form would be *[ˌhiranˈkiran] if reduplication occured after the word level. For form (g) I hypothesize the underlying form /laluv/, which loses its final consonant at the word level in isolation. Therefore if it underwent typical late reduplication the expected output would be *[ˌlaluˈlalu]. Instead the obstruent /v/ survives to coalesce with [l]$_w$, producing [d].

3.11.2.10 *Prefixed Forms*

There is one other category of forms that requires some attention: reduplication with prefixes. Here I will confine myself to the active prefix *an-*, but others (*e.g.,* present *m-* when combined with active prefix ∅-) behave in the same manner. Table (265) shows a number of examples.

(265)

Gloss	Simple	Present Active	Redup.	Redup. Pres. Act.
(a) *cut, read*	ˈvaki	maˈmaki	ˌvakiˈvaki	maˌmakiˈvaki
(b) *writing*	ˈsuratʃa	maˈnuratʃa	ˌsuraˈtsuratʃa	maˌnuraˈtsuratʃa
(c) *shiver*	ˈhuvitʃa	maŋˈguvitʃa	ˌhuviˈkuvitʃa	maŋˌguviŋˈguvitʃa
(d) *refusal*	ˈla	manˈda	ˌlaˈla	manˌdanˈda
(e) *hit, kill*	ˈvun	maˈmun	ˌvunuˈvun	maˌmunuˈmun/ maˌmunuˈvun
(f) *lie*	ˈlɑ͡iŋga	maŋˈdɑ͡iŋga	ˌlɑ͡iŋgaˈlɑ͡iŋga	manˌdɑ͡iŋganˈdɑ͡iŋga/ manˌdɑ͡iŋgaˈlɑ͡iŋga

In the first group, the final nasal of the prefix is not copied during reduplication. In the second group, it is. In the third group, it is optionally copied (according to Keenan & Razafimamonjy (1998), children are more likely to choose to copy the nasal, adults less likely). Two analyses are possible for these facts: either the nasal-copying forms represent cases where the present active form has lexicalized (so the input is actually, *e.g.*, /m+anurat+RED/ or /manurat+RED/ rather than /m+an+surat+RED/), or reduplication is realized in the nasal-copying cases by doubling the prefix along with the base. The first of these analyses seems to me more plausible.

In all the cases of free variation treated in this analysis, I hypothesize that the phenomenon is due to uncertainty on the part of the speaker between competing underlying forms.

3.11.2.11 *Summary*

Thus the final story for reduplication is that many (perhaps most) forms are joined after the word level, the base (with any prefixes) and the reduplicant (with any suffixes) having proceeded in isolation up to that point. The remaining reduplicated forms proceed in isolation either not at all, or at most through the stem level. Given the stress pattern of [ˌlaludaˈluvan], it is simplest to say that these early-reduplicated forms proceed in isolation through the stem level and join just before the word level.

3.11.3 RANKINGS FOR REDUPLICATION

The great benefit of the chosen analysis is that, as will be shown below, almost the entire picture of reduplication falls out from the existing rankings. The only remaining question is how to account for the shape of the reduplicant. In order to determine that, I will first discuss the particulars of how a reduplicating form is represented, and then I will introduce the necessary rankings by looking at a series of examples.

3.11.3.1 *Representation*

This analysis treats the reduplicant as a form similar to the base. That is, its underlying form is the same as the underlying form of the base, except that in the underlying form the reduplicant is marked as being in the morpheme RED and the base is marked BAS[35]. There is a constraint family MAXIB which acts exactly like the MAXIO family except that it does not apply inside the reduplicant, and MAXIR, which applies only inside the reduplicant.

[35] Nothing in the analysis depends on this—the base could possibly remain unmarked.

Similarly, any constraint may be restricted to apply only inside the reduplicant, or only outside of it.

3.11.3.2 *Examples*

The following examples show reduplication where the base has more than two syllables, with penultimate main stress at the stem level.

(266)

Gloss	Simple	Reduplicated	Hyp. Red. Output (Stem)
(a) *forget*	haˈdin	haˌdinuˈdin	[ˈdinu]$_s$
(b) *sadness*	ˌalaˈhelu	ˌalaˌheluˈhelu	[ˈhelu]$_s$
(c) *verandah*	ˌlavaˈraŋgan	ˌlavaˌraŋganˈd͡raŋgan	[ˈraŋgan]$_s$

Here the reduplicant consists of the last two syllables of the root. There are a number of factors to account for:

1. Why only two syllables?
2. Why the final two syllables?

A possible answer is as follows:

1. The syllable that carries the greatest stress in a word is the most prominent, and therefore it should be copied.
2. The right edge of the word is prominent as well, and therefore should be copied.
3. More than this should not be copied in order to limit overall word length, or, possibly, to cope with the difficulty of copying.

To encode this answer, I employ the following constraints:

(267) MaxIR:

Output a violation for any underlying segment of the morpheme marked RED *that has not surface correspondent.* Here this constraint has the effect of making the reduplicant as large as possible.

(268) *M([+stress,-prim], red):

Short form—*[+stress,-prim]/$_{\text{RED}}$[_]. *Output a violation for each instance of a secondary stressed segment at the surface level within the reduplicant.* Here this undominated constraint has the effect of eliminating any secondary stressed vowels from the reduplicant. Since a word can have only one primary stressed vowel (via *[+prim] (24), *q.v.*), this means the reduplicant will have only one stressed vowel.

(269) *InitialM([-stress,V], C, red):

Short form—*[-stress,V]/$_{\text{RED}}$[C$_0$_. *Output a violation if the initial vowel of the reduplicant is unstressed.* This undominated constraint has the effect (when combined with the previous constraint) of forcing the reduplicant to begin with a primary stressed vowel.

(270) *Final([+stress,-prim], [-stress]):

Short form—*Rtmost2ndary. *Output a violation if the rightmost stress of a word is* [-prim]. To satisfy this constraint, a word must either have no stress at all, or the rightmost stress must be primary. Any other stressed in the word may be secondary or primary, as far as the constraint is concerned. Undominated.

(271) *([+PRIM]):

Output a violation for each primary-stressed vowel in a word. This, along with *RT-MOST2NDARY and EXISTS([+STRESS]) has the effect of the standard constraint CULMINATIVITY plus rightward alignment of primary stress.

(272) EXISTS([+STRESS]):

Output a violation for any word that has no stress. Undominated. This ensures, in addition to the obvious requirement that all words carry stress, that the reduplicant will not be empty, since it is a separate word in the earlier levels.

(273) PRESERVERTM(X, RED):

Short form—PRESERVERT. See §3.2.3.2 for a detailed explanation. *Ensure that the part of the underlying base that is preserved in the stem-level output form of the reduplicant is contiguous and rightward aligned.* This constraint is vacuously satisfied by an empty reduplicant, but this outcome is prevented by EXISTS([+STRESS]).

Of these constraints, most are undominated: all but MAXIR and *[+PRIM]. One ranking is already known (see §3.4.2 above):

(274) *RTMOST2NDARY ≫ *[+PRIM]

Many others can be inferred from the output [ˌalaˌheluˈhelu].

(275) /alahelu+RED/ → [ˌala'helu]$_s$+['helu]$_s$ 'sadness+*red.*'

/$_{RED}$[alahelu]/	*RtMost2ndary *[+stress,-prim]/$_{RED}$[_]	PreserveRt	*[-stress,V]/$_{RED}$[C_0_	*FinalLapse	*[+prim]	MaxIR(C)	*Lapse
☞ ØØØ'helu					*	*	
(1) 'ala'helu					**!		
(2) Ø'lahelu				*!	*		*
(3) Øla'helu			*!		*		
(4) Ø1Ø'helu		*!			*		
(5) Ø'lahØlu		*!			*		
(6) Ø1'ahelØ		*!			*		
(7) ˌala'helu	*!				*		
(8) 'alaˌhelu	*!	*			*		

Tableau (275) illustrates how the reduplication template constraints above limit the reduplicant to the rightmost foot in base. Note that MaxIR(C) appears here instead of the bare MaxIR. This is because Exists([+stress]) and PreserveRt, together with the stress templates, serve to make MaxIR(V) superfluous. The constraint MaxIR(V) may exist, but it is inactive in the ranking.

The first, seventh, and eighth losing candidates in the tableau show total reduplications. The seventh is, in fact, a perfect copy of the base. It, as well as the eighth, lose out because secondary stresses are forbidden in the reduplicant.

(276) *[+stress,-prim]/$_{RED}$[_] $\gg_s$ MaxIR(C)

The eighth candidate violates *RTMOST2NDARY as well. The first losing candidate avoids these pitfalls, but has more than one primary stress.

(277) *[+PRIM] ≫MAXIR(C)

The second losing candidate shows that the normal stress pattern of the language helps to determine the size of the reduplicant. If the primary stress is moved back by one syllable, it creates a stress lapse, and either *FINALLAPSE or *LAPSE therefore rules it out.

The third losing candidate illustrates the requirement that the reduplicant must begin with a stressed syllable.

(278) *[-STRESS,V]/$_{\text{RED}}$[C_{0}_ ≫$_{s}$ MAXIR(C)

Candidates (4)–(6) show how PRESERVERT rules enforces contiguity and rightward alignment.

(279) PRESERVERT ≫$_{s}$ MAXIR(C)

At this point all that remains is to examine monosyllabic reduplicants.

(280)

Gloss	Simple	Reduplicated	Proposed RED Output (Stem)
(a) *again*	in'd͡rai͡	in,d͡rai͡n'd͡rai͡	[n'd͡rai͡]$_{s}$
(b) *bark*	vu'vu	vu,vu'vu	['vu]$_{s}$
(c) *big*	,lehi'be	,lehi,be'be	['be]$_{s}$

Table (280) can be summarized by noting that in all cases where the isolation form of a root has final stress, the reduplicant consists of a single syllable (more precisely, as shown by the inclusion of [n] in (a), it consists of the final stressed vowel segment, preceded and followed by any consonants that were contiguous to it in the isolation form). The same undominated constraints (IDENTIO([+STRESS]/_C_0# and WSP) that forced final stress in the isolation form force one-syllable reduplicants. See ranking (281) and tableau (282).

(281) WSP, IDENTIO([+STRESS])/_C_0# $\gg_s$ MAXIR(C)

(282) /lehi'be+RED/ → [ˌlehi'be]$_s$+['be]$_s$ 'big+*red.*'

/$_{\text{RED}}$[lehi'be]/	IDENTIO([+STRESS])/_C_0#	MAXIR(C)	NONFINALITY	IDENTIO([+STRESS])
☞ ØØØØ'be		**	*	
ØØ'hibe	*!	*		*

3.11.4 SUMMARY

Perhaps this is not the case for all languages, but the somewhat unusual patterns of Malagasy reduplication turn out not to require the complex machinery afforded by Base Reduplicant Correspondence Theory (in fact, that machinery is insufficient for the Malagasy patterns). Instead, the patterns fall out from the basic LPM-OT analysis that covers the rest of the language. The only additions required are reduplicant template constraints and the idea that reduplication is compounding of a root together with itself.

3.12 CONCLUSION

In reviewing this analysis, three features, aside from exhaustiveness, emerge as worthy of note: it uses a non-correspondence analysis of reduplication; it is overwhelmingly comprised of undominated constraints; it utilizes a system of constraint families that differs somewhat from those used elsewhere; and it uses LPM-OT extensively to account for opaque phenomena.

3.12.1 COMPOUNDING ANALYSIS OF REDUPLICATION

The primary benefit, to my mind, of a non-correspondence treatment of reduplication (leaving aside the obvious benefit of covering the data more completely in this case) is computational in nature. Chapter 2 shows that a computational model of Correspondence Theory reduplication must necessarily be of greater time complexity than a model that does not employ Base-Reduplicant correspondence. This increased complexity has the practical result that for such a model to be feasibly computable by a finite device such as the human brain, either B-R correspondence constraints must be quite low in the ranking[36] or the size of the base of reduplication must be limited. If a Correspondence Theory account of reduplication is indeed necessary for the rare cases of overapplication, then one should expect that one of these mitigations will apply. It would be a valuable research program to construct whole-phonology analyses of the languages (such as Malay

[36]Low-ranking of B-R correspondence constraints (actually what is relevant is the ranking level of the highest-ranked B-R correspondence constraint) only helps with a weighted FSM model rather than a transducer model, but then again a full model of reduplicative correspondence theory seems not to be possible with a transducer model.

and Javanese) in which a B-R correspondence account is claimed to be necessary.

3.12.2 DOMINANCE OF THE UNDOMINATED

A perhaps surprising fact about the Malagasy analysis is that three quarters of the constraints that are active in the analysis (*i.e.*, shown to be necessary by the ranking algorithm) are exceptionlessly true at their level, or in the usual parlance, undominated[37]. What implications does this fact have for the Optimality Theoretic claim that all constraints are innate and only the rankings are learned? The first question that might be asked from this point of view is whether many of these constraints are universally undominated. Such a state of affairs would strengthen the argument for universal constraints and make matters easier for ranking algorithms. Unfortunately, however, this appears not to be the case. There is no reason to believe that any of the active undominated constraints in this analysis could not be violable in some language or another. Instead, the lesson I take from this state of affairs is that most active constraints in a given analysis are exceptionlessly true and only a small minority submit to the necessity of constraint demotion, but the learner cannot assume inviolability for any constraint. The type of learning algorithm that stands to benefit from this situation is not the usual Optimality Theoretic ranking-only algorithm, but rather one in which the constraints themselves are learned. In such an algorithm the ranking step would be reserved for non-universal generalizations. An exceptionless generalization should be easier to learn than one that only applies to a subset of the data.

[37] The constraints that are undominated in the post-lexical level are surface-true, and are similarly undominated at lower levels. At each lower level additional constraints are level-true, or undominated at that level.

3.12.3 CONSTRAINT SYSTEM

The constraint system used here (described in §3.2.3) was arrived at by a process of attempting as much as possible to use standard constraints of the sort seen extensively in the Optimality Theoretic literature. In some cases, however, this was not possible due either to the excessive power of those constraints or to the unforeseen side-effects some of them have (as in the case of contextual faithfulness, discussed below). For example, the ALIGN family of constraints proved unimplementable in general. Various *SEQUENCE, *FINALSEQUENCE, *INITIAL, *FINAL, etc. constraints were used instead. Another example is the general IDENTIO constraint which penalizes any feature difference between an underlying segment and its surface correspondent. Actual translation of this constraint into a weighted finite state machine would be prohibitively expensive in terms of the size of the resulting machine, so I did not try to use this constraint.

Another area where I was unable to use standard constraints is contextual faithfulness. Typically in the literature the context for a contextual faithfulness constraint is expressed on the surface. For example, DEPIO(C)/_V would penalize insertion of a consonant before a surface vowel. Such constraints are perfectly easy to implement, but when employed in a grammar they make a correct ranking rather tricky to find, because such constraints act simultaneously as a constraint of faithfulness to an underlying segment and, perhaps unintentionally, as a well-formedness constraint banning the context. In the example of DEPIO(C)/_V, the constraint system must ensure that GEN does not delete a following vowel in order to be able to insert a consonant. For example, while developing this analysis I had a constraint like MAXIO(V)/_# where the word-final context was surface rather

than underlying. Given an input like /be/ where /e/ was a dispreferred vowel, the output was [ba] where the vowel [a] had been inserted word-finally in order to be able to delete /e/. This sort of thing seems less than natural to me, but frequently occurs with surface contextual faithfulness. A constraint system where both the object to which the constraint is mandating faithfulness and the context where that faithfulness is enhanced are underlying is much better behaved. An extension of underlying contextual faithfulness may be required, however: *realized* underlying contextual faithfulness. That is, faithfulness in the environment of some entity that is both underlying and surface. For example, MaxIO(C)/_V:V would be faithfulness to an underlying consonant that precedes a realized underlying vowel. An example requiring this sort of faithfulness constraint is given by Wilson (2000) (there arguing for an entirely different constraint scheme)—a language with syncope and consonant cluster simplification where underlying /akta/ appears on the surface as [ata], not *[aka], and, further, underlying /akata/ appears as [ata] as well. The point of this example is that the rightmost consonant is always chosen here, even though hypothetical *[aka] satisfies MaxIO(C)/_surface(V) just as well in both examples, and MaxIO(C)/_ur(V) in the second example. In both cases, however, MaxIO(C)/_V:V makes the correct prediction.

Finally, most surface well-formedness constraints in the literature appear to be *ad hoc,* whereas here there is a standard scheme for representing them: all well-formedness constraints fall into the families *(), *Sequence, *Contour, and Exists, with contextual variants (word-initial, word-final, inside a morpheme, morpheme-initial, morpheme-final).

3.12.4 LPM-OT

The final analysis characteristic I would like to discuss is its use of LPM-OT, in three contexts: how it interacts with Correspondence Theory, what it implies for learnability, and how it compares with other treatments of opaque phonology.

3.12.4.1 *Compatibility with Correspondence Theory*

There is no reason why LPM-OT should not be compatible with a Correspondence Theory account of reduplication, but adding LPM-OT does complicate a Correspondence Theory account somewhat. If evidence of stem-level morphological boundaries is to be erased at the word level and word-level boundaries are to be erased at the post-lexical level, then Base-Reduplicant correspondence constraints are only active at the level of which RED is an affix, so if RED is a stem-level affix, B-R correspondence will take place there. On the other hand, perhaps it will be necessary to preserve some morphological information from one level to another.

3.12.4.2 *Learnability Implications*

There are two major differences between an LPM-OT grammar and a grammar used in one of the other Optimality Theoretic frameworks that have been designed for analyzing phonological opacity: (1) an LPM-OT grammar has three constraint rankings instead of just one, and (2) an LPM-OT grammar uses only the constraints of standard Optimality Theory with no additional constraints specific to treatment of opacity. The first of these differences is a problem for learnability, but the second is a major boon. It remains to

be seen whether the problematic aspect outweighs the beneficial, but tentatively I would assert that the advantages LPM-OT presents for learnability outweigh the disadvantages. I must be tentative at this point, of course, because no concrete algorithm has been proposed for learning opaque phonology under *any* of the frameworks. Here, at least, is a sketch of an algorithm that might work for LPM-OT:

Post-lexical phonology is learned first, using no knowledge of morphological structure—this learning process could use a phonotactic learner of the sort discussed in Albro 2000, in addition to other facilities. The rankings of lower levels (stem, word) then differ from those of the post-lexical level due to the morphological knowledge that can be applied to lower levels or increased regularities that can be found when affixes are removed or phonological changes occurring at the post-lexical level are reversed[38]. For example, at the post-lexical level the stress pattern of Malagasy is fairly idiosyncratic. WSP and *EXTLAPSE are the only constraints that appear to apply without exception at the top level. Once final epenthesis is removed, however, it becomes clear that *FINALLAPSE can be promoted to undominated at the word level, and the basic right-to-left trochaic stress pattern becomes clear at the stem level once the agency of suffix-driven stress shift becomes clear. Thus for stress the overall story is one of increasingly powerful markedness constraints as you move from the post-lexical level down to the stem level, reflecting generalizations that become apparent as the details of morphology are used in the learning process.

The following principles can be used to guide analysis or learning when levels are

[38] Reversing phonological changes is trivial with a transducer model of Optimality Theory, but even with a non-transducer model the post-lexical learner would include as its output surface-underlying pairs for the utterances that formed its input. The underlying forms in these pairs, then, are the result of reversing post-lexical phonology.

involved:

- "Lexical" contrasts between stem groups which seem to have similar phonological properties may diagnose a higher ranking of some markedness constraint at level n than at level $n-1$, this higher ranking having obscured an underlying distinction (as in the /n#/ vs. /ŋ#/ case here).

- Paradigm-Uniformity/Base-Exponence-type effects diagnose an increasing rôle for faithfulness constraints at some level n when compared with $n-1$, as in Malagasy stress.

- Any segment that seems invisible to a phenomenon is introduced at a higher level than that phenomenon (as in Malagasy final epenthetic vowels vs. the trochaic stress pattern).

- Cases where some phenomenon appears influenced by a segment that does not exist indicate segment deletion at a higher level than the locus of the phenomenon (as in Malagasy post-nasal vowel deletion).

3.12.4.3 *Opaque Phonology Frameworks Compared*

Every framework that is capable of accounting for opaque phonological phenomena, including LPM-OT, is more complex than standard Optimality Theory on some standpoint, but they differ widely on just how much they differ from it. I must note beforehand that I cannot give an explicit complexity measure for any of these frameworks. Computational complexity depends on a specific algorithm and a specific representational scheme.

In general, though, I can say that the complexity of generation using any of the finite state methods is dependent on the alphabet size of the representation and on the number of states in the machines that are being manipulated. Any framework that increases the number of tiers or levels of representation required for generation necessarily increases one or both of these factors (alphabet size or number of states), generally in an exponential fashion, since, for one thing, GEN must encode all possible correspondences between the different levels of representation.

LPM-OT The LPM-PT model of phonological opacity fares well on the scale of computational complexity. In terms of the time complexity of generation, it differs from the standard non-opaque model of Optimality Theory by no more than a constant factor. This project has, I believe, shown it to be within the realm of the feasible. It does present some difficulties for learnability, however, but opacity is, I would think, inherently more difficult to learn, and LPM-OT does have the advantage that it introduces no new constraint families and no complex apparatus.

Note, however, that LPM-OT is not without its flaws. It predicts a level ordering model where stem-level affixes are universally farther from word-boundaries than word-level affixes. To the extent that this is not true, LPM-OT is inadequate without some emendation.

BASE EXPONENCE Base Exponence (Kenstowicz 1995) was one of the first frameworks proposed for dealing with opaque phenomena in Optimality Theory. It includes the usual Input-Output correspondence and also Output-Output correspondence between an affixed form and its stem. Since this involves an extra level of representation over standard

Optimality Theory, this framework has greater time complexity of generation with respect to LPM-OT, although not overwhelmingly greater. There are certain phenomena, however, such as the dual-stem correspondence shown by Malagasy reduplication (*e.g.,* a form such as [ˌavuʧaˈvutan] seems to correspond both with the isolation form [ˈavuʧa] and the suffixed form [aˈvutan]), for which the framework has no ready explanation.

PARADIGM UNIFORMITY/TRANSDERIVATIONAL CORRESPONDENCE The frameworks of Paradigm Uniformity (Steriade 1996; Flemming 1995) and Transderivational Correspondence (Benua 2000) appear to cover opacity data fairly well. These are frameworks in which an output form is faithful not only to the underlying form, but also to all other members of its paradigm. This implies that its representations will therefore have many more levels of representation than LPM-PT does, and therefore the output of GEN will be exponentially larger. Thus, these frameworks will necessarily have exponentially greater time complexity of generation. See Albro 1998a for a sketch of how one might formally model these frameworks in the weighted finite state paradigm.

TWO-LEVEL CONSTRAINTS Some opaque phenomena may be analyzed by reference to constraints which act essentially as markedness constraints that have reference to the underlying form (Koskenniemi 1983). The use of two-level constraints has been criticized, for example by Kiparsky (2000), as tending to miss available empirical generalizations. From the perspective of computational complexity, however, two-level constraints do not add much, if any, complexity (the only complexity they might add is by virtue of the fact that the finite-state representations of two-level constraints tend to require a larger number of

states and edges than other constraints) to the standard model. There are somewhat less minor learnability implications, however, in the larger constraint space that a learner must traverse. Note that proposals to locally conjoin faithfulness and markedness constraints can be characterized similarly.

Sympathy Theory Sympathy Theory (McCarthy 1998) has a similar complexity to Paradigm Uniformity and Base Exponence. In this framework output forms may be in correspondence with forms that are produced by varying constraint rankings. If no more than one sympathy candidate is employed by an analysis, then the computational complexity will be similar to Base Exponence, but each additional sympathy candidate will increase the complexity exponentially.

3.12.5 summary

This analysis of Malagasy has served to show the sort of insights that might be gained from a whole-language analysis. It exemplifies what such an analysis might look like, points out flaws in the Correspondence Theory model of reduplication, and demonstrates the utility of the lpm-ot framework. It is hoped that others will soon employ the tools that made this analysis possible, or others like them, to begin construction of a large body of such analyses so that phonologists will be able to construct a more complete picture of what the languages of the world are really like.

APPENDIX A

The Weighted Finite State Model of Optimality Theory

The basic structure of Optimality Theory comprises three components: GEN, a function which converts an underlying representation (a single string) into a representation of the infinite set of possible outputs (for I-O Correspondence Theory the outputs include the input, plus a correspondence relation between the segments of the input and the outputs); CON, a universal set of constraints, which map output candidates to integer penalty values according to some constraint-specific metric; and EVAL, an function that uses the constraints, arranged in a language-specific ranking, to select from the infinite set of potential outputs produced by GEN a single "most harmonic" candidate.

A.1 GEN

The GEN function, in the Ellison model, is fairly simple. It simply produces a finite state machine representing the input string paired with all well-formed output strings (the set of well-formed output strings is the primary difference between the Ellison, Eisner, and Albro models[1]).

[1] The Albro model here primarily refers to the model used in this dissertation rather than the separate Albro model from Albro 1998a, which uses a representation similar to that of Eisner but modified to be complex and powerful enough for full-scale real-world phonological analyses.

A.2 CON

Each constraint in CON is implemented as a weighted finite state machine of a particular sort. For every constraint c, the implementation C of c is a weighted finite state machine, *i.e.,* a 5-tuple $\langle Q, \Sigma, \delta, q_0, A\rangle$:

- $Q \subseteq \mathbb{N}$ is a set of states.
- Σ is a set, disjoint from Q, of symbols making up the alphabet of the machine.
- δ is the transition function. It maps from $Q \times (\Sigma \cup \epsilon)$ to $2^{Q\times\{0,1\}}$.
- $q_0 \in Q$ is the initial state of the machine.
- $A \subseteq Q$ is a set of acceptor (final states).

The extended transition function $\hat{\delta}$ is defined as follows: $(s, \omega) \in \hat{\delta}(q, u)$ iff $(s, \omega) = (q, 0)$ and $u = \epsilon$, or there exist $u_i \in (\Sigma \cup \epsilon)$ and $\omega_i \in \{0, 1\}$, for $1 \leq i \leq n$ and $n \geq 1$, such that $u_1u_2 \ldots u_n = u$ and $\Sigma_{i=1}^{n}\omega_i = \omega$, $(s_i, \omega_i) \in \delta(s_{i-1}, u_i)$ for $1 \leq i \leq n, q = s_0$ and $s = s_n$. The language accepted by a constraint implementation C is the set $L(C) = \{u|\hat{\delta}(q_0, u) \cap (A \times \{0, 1\}) \neq \emptyset\}$. For a constraint, $L(C) = \Sigma^*$. For a given string u the weight $W_C(u)$ assigned u is defined as $\min\{\omega|(q, \omega) \in \hat{\delta}(q_0, u), q \in A\}$. The constraint c itself is then the function W_C—a total function from Σ^* to $\mathbb{N}$.

A.3 EVAL

My FSM algorithm for the EVAL function operates as follows[2]:

1. GEN applies to the input string, producing an FSM, which will be referred to as *cands*.

2. *cst* is set to the most highly ranked constraint.

3. *cands* is intersected with *cst* to yield the strings of *cands,* each weighted with a penalty value.

4. A modified form of Dijkstra's Single Source Shortest Paths algorithm (see Albro 1998a) is used to remove all but the least penalized candidates. The variable *cands* now refers to the result of this step.

5. *cst* is set to the next most highly ranked constraint. If there is none, the value of EVAL is set to the current value of *cands* and the algorithm exits. Otherwise the algorithm continues at step 3.

[2]Eisner's version, on which my version is based, is the same for the most part. Ellison's version uses a different intersection algorithm.

APPENDIX B

MCFG Background

An MCFG is an extension to context free grammars that yields greater expressive power (an MCFG can describe any language that a CFG can, but there are languages that a context sensitive grammar can describe that an MCFG cannot). In a context free grammar, each non-terminal symbol in the grammar represents (that is, *yields*) a set of strings. A multiple context free grammar expands upon this by allowing non-terminal symbols to represent *tuples* (that is, groupings) of sets of strings. For example, in a CFG the non-terminal N might yield the set {"dog," "cat," "mouse," "ball"}. In an MCFG NV might yield the tuple ⟨{ "dog," "cat," "mouse"}, { "eats," "sleeps," "drinks"}⟩. Here is a formal definition of MCFGs. This is followed by an exploration of their descriptive power, some examples, and finally a definition of the intersection of an MCFG with an FSM.

B.1 DEFINITIONS

B.1.1 MCFG

An MCFG G is defined as a 5-tuple $\langle N, \Sigma, F, P, S \rangle$where

- N is a set of nonterminal symbols. Each $A \in N$ has an associated *degree* $d(A) \in \mathbb{N}$

- Σ is a set of terminal symbols, disjoint from N.
- F is a set of functions $f \in F$, each f being a function from $(\Sigma^*)^{d_1(f)} \times (\Sigma^*)^{d_2(f)} \times \cdots \times (\Sigma^*)^{d_{a(f)}(f)}$ to $(\Sigma^*)^{r(f)}$ where:
  - $a(f)$ describes the number of arguments of f
  - $r(f)$ describes the range of f
  - $d_i(f)$, $1 \leq i \leq a(f)$ is the degree of each argument of f
- P is the set of *productions* of the grammar, where each $p \in P$ is a finite subset of $\bigcup_q (F_q \times N^{q+1})$ where F_q is the subset $\{f \in F | a(f) = q\}$. A production is conventionally notated as $A_0 \rightarrow f[A_1, A_2, \ldots, A_{a(f)}]$. For any such production, $r(f) = d(A_0)$, $d_i(f) = d(A_i)(1 \leq i \leq a(f))$.
- $S \in N$ is the *start symbol* for the grammar. The grammar must be such that $d(S) = 1$.

The following requirements also pertain to the members of F:

Let $\overline{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id_i(f)})$ and $X = \{x_{ij} | 1 \leq i \leq a(f), 1 \leq j \leq d_i(f)\}$. Then if $f \in F$ is *terminating* ($a(f) = 0$), then $r(f) = 1$, and f is defined as $f = \alpha$ for some $\alpha \in \Sigma^*$. If f is *nonterminating* ($a(f) > 0$), then each component $f^h (1 \leq h \leq r(f))$ of f is defined as $f^h[\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_{a(f)}] = z_{h1}, z_{h2}, \ldots, z_{h\nu_h(f)}$ where $z_{hk} \in X$ $(1 \leq k \leq \nu_h(f))$, subject to the condition that each member of X appears exactly once in the definition of f.

B.1.2 GRAMMAR ARITY

$d(G)$, the arity of a grammar, is defined as follows: $d(G) = \max\{d(A)|A \in N\}$. An MCFG G with $d(G) = m$ is referred to as an m-MCFG.

B.1.3 LANGUAGE/YIELD SET

The language (yield set) $L_G(A)$ of a nonterminal category A within an MCFG G is defined to be the smallest set satisfying the following conditions (here θ is a tuple $\langle \alpha_1, \alpha_2, \ldots, \alpha_{|\theta|} \rangle$, $\alpha_i \in \Sigma^*$):

1. If a terminating rule $A \to \theta \in P$, then $\theta \in L_G(A)$
2. If $\theta_i \in L_G$ $(1 \leq i \leq a(f))$ for $A \to f[A_1, A_2, \ldots, A_{a(f)}] \in P$ then $f[\theta_1, \theta_2, \ldots, \theta_{a(f)}] \in L_G(A)$

Then $L(G) = L_G(S)$.

B.1.4 DERIVATION TREE

For an MCFG G, the set $CL(G)$ contains the derivations trees for all strings in G. This is defined as $\lim_{k\to\infty} CL^k(G)$ where $CL^k(G)$ is defined as follows:

1. For a terminating rule $A \to \theta$, the tree consisting of a single node labeled $A : \theta$ is a derivation tree of θ, and is a member of $CL^0(G)$.
2. For all $\tau \in CL^{k-1}(G)$, $\tau \in CL^k(G)$.

3. Given a rule $A_0 \to f[A_1, A_2, \ldots, A_{a(f)}] \in P$, if for all i, $1 \leq i \leq a(f)$, $\tau_i \in CL^{k-1}(G)$ is a derivation tree of θ_i whose root is labeled $A_i : \theta_i$, then the derivation tree of $\theta_0 = f[\theta_1, \theta_2, \ldots, \theta_{a(f)}]$ is the tree whose root is a node labeled $A_0 : \theta_0$ that has $a(f)$ children, of which child i for all i such that $1 \leq i \leq a(f)$ is isomorphic to τ_i. This tree is a member of $CL^k(G)$.

4. $CL^k(G)$ has no other members.

B.2 EQUIVALENCES AND PLACEMENT WITHIN THE CHOMSKY HIERARCHY

An MCFG is a specialization of Pollard's (1984) Generalized Context-Free Grammars. The complexity of the languages describable by MCFGs is placed in the Chomsky hierarchy is as follows:

$$RL \subsetneq CFL = 1 - MCFL \subsetneq HL = MHL = TAL = LIL = CCL \subsetneq 2 - MCFL \subsetneq \ldots \subsetneq MCFL = LCFRS = ML \subsetneq PMCFL \subsetneq CSL,$$

where these are defined as follows:

RL Regular Language (describable by a Finite State Machine)

CFL Context Free Language

1-MCFL 1-ary Multiple Context Free Language (see §B.1.2)

HL Head Language (Pollard 1984)

MHL Modified Head Language (Vijay-Shanker *et al.* 1986)

TAL Tree-Adjoining Language (Joshi *et al.* 1975)

LIL Linear Indexed Language (Gazdar 1985; Hopcroft & Ullman 1979)

CCL Combinatory Categorial Language (Steedman 1986; Steedman 1987)

m-MCFL m-ary Multiple Context-Free Language

MCFL $\lim_{m\to\infty}$ m-ary Multiple Context-Free Language

LCFRS Linear Context-Free Rewrite System (Vijay-Shanker *et al.* 1987)

ML Minimalist Language (Stabler 1997)

PMCFL Parallel Multiple Context-Free Language

CSL Context Sensitive Language

B.3 EXAMPLES

B.3.1 COPY LANGUAGE

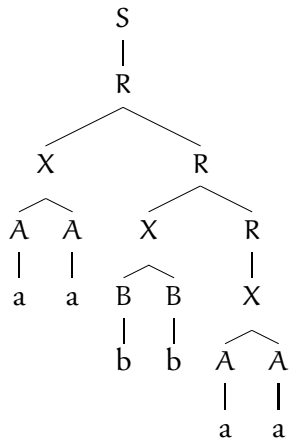
The following grammar yields the language $\{ww|w \in \{a, b\}^*\}$:

$$S \rightarrow f_1[R], f_1[\overline{x}_1] = x_{11}x_{12}$$

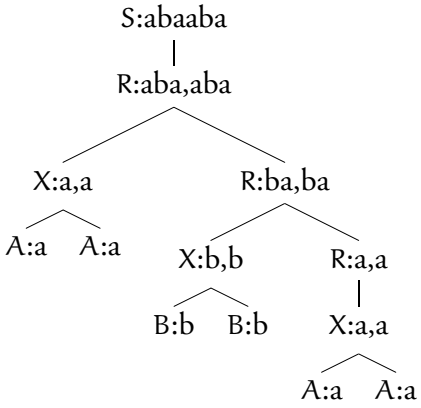
$$S \rightarrow \epsilon$$

$$
\begin{aligned}
R &\rightarrow f_2[X], f_2[\overline{x}_1] = x_{11}, x_{12} \\
R &\rightarrow f_3[X, R], f_3[\overline{x}_1, \overline{x}_2] = x_{11}x_{21}, x_{12}x_{22} \\
X &\rightarrow f_4[A, A], f_4[\overline{x}_1, \overline{x}_2] = x_{11}, x_{21} \\
X &\rightarrow f_4[B, B] \\
A &\rightarrow a \\
B &\rightarrow b
\end{aligned}
$$

An example derivation is the string abaaba, whose derivation tree (not the one according to the definition, but a derivation tree more like the usual one for a CFG) appears as follows:



It is immediately apparent from this tree what the difference is between an MCFG and a CFG—the leaves of this tree, taken in order, do not produce the derived string. Instead, a function applies at each node to change the order of words within the string. The type of derivation tree given in the definition in §B.1.4 makes this more clear:



Here each node has been decorated with the string that was derived from its category. The derivation might be put into words as follows:

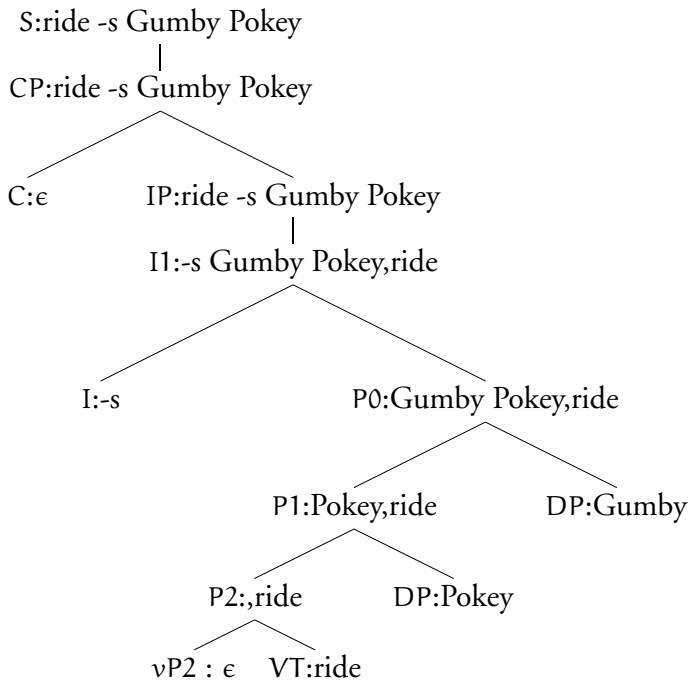
1. Rule $\mathtt{A} \to \mathtt{a}$ applies twice; $\mathtt{a} \in \mathrm{L}_{\mathrm{G}}(\mathtt{A})$.
2. Rule $\mathtt{X} \to \mathtt{f}_4[\mathtt{A}, \mathtt{A}]$ applies; $\mathtt{f}_4[\mathtt{a}, \mathtt{a}] = \mathtt{a}, \mathtt{a} \in \mathrm{L}_{\mathrm{G}}(\mathtt{X})$.
3. Rule $\mathtt{R} \to \mathtt{f}_2[\mathtt{X}]$ applies; $\mathtt{f}_2[(\mathtt{a}, \mathtt{a})] = \mathtt{a}, \mathtt{a} \in \mathrm{L}_{\mathrm{G}}(\mathtt{R})$.
4. Rule $\mathtt{B} \to \mathtt{b}$ applies twice; $\mathtt{b} \in \mathrm{L}_{\mathrm{G}}(\mathtt{B})$.
5. Rule $\mathtt{X} \to \mathtt{f}_4[\mathtt{B}, \mathtt{B}]$ applies; $\mathtt{f}_4[\mathtt{b}, \mathtt{b}] = \mathtt{b}, \mathtt{b} \in \mathrm{L}_{\mathrm{G}}(\mathtt{X})$.
6. Rule $\mathtt{R} \to \mathtt{f}_3[\mathtt{X}, \mathtt{R}]$ applies; $\mathtt{f}_3[(\mathtt{b}, \mathtt{b}), (\mathtt{a}, \mathtt{a})] = \mathtt{ba}, \mathtt{ba} \in \mathrm{L}_{\mathrm{G}}(\mathtt{R})$.
7. Rule $\mathtt{A} \to \mathtt{a}$ applies twice.
8. Rule $\mathtt{X} \to \mathtt{f}_4[\mathtt{A}, \mathtt{A}]$ applies.
9. Rule $\mathtt{R} \to \mathtt{f}_3[\mathtt{X}, \mathtt{R}]$ applies; $\mathtt{f}_3[(\mathtt{a}, \mathtt{a}), (\mathtt{ba}, \mathtt{ba})] = \mathtt{aba}, \mathtt{aba} \in \mathrm{L}_{\mathrm{G}}(\mathtt{R})$.
10. Rule $\mathtt{S} \to \mathtt{f}_1[\mathtt{R}]$ applies; $\mathtt{f}_1[\mathtt{aba}, \mathtt{aba}] = \mathtt{abaaba} \in \mathrm{L}_{\mathrm{G}}(\mathtt{S}) = \mathrm{L}(\mathtt{G})$.

B.3.2 VISO

The following grammar, when suitably extended, yields a language with Verb Infl Subject Object order, including such sentences as "wave -s Gumby" and "ride -s Gumby Pokey":

$$
\begin{aligned}
S &\rightarrow f_1[CP], f_1[\overline{x}_1] = x_{11} \\
CP &\rightarrow f_2[C, IP], f_2[\overline{x}_1, \overline{x}_2] = x_{11}x_{21} \\
IP &\rightarrow f_3[I1], f_3[\overline{x}_1] = x_{12}x_{11} \\
P0 &\rightarrow f_4[P1, DP], f_4[\overline{x}_1, \overline{x}_2] = x_{21}x_{11}, x_{12} \\
I1 &\rightarrow f_5[I, P0], f_5[\overline{x}_1, \overline{x}_2] = x_{11}x_{21}, x_{22} \\
P1 &\rightarrow f_6[\nu P, VI], f_6[\overline{x}_1, \overline{x}_2] = x_{11}, x_{21} \\
P1 &\rightarrow f_4[P2, DP] \\
P2 &\rightarrow f_6[\nu P2, VT] \\
DP &\rightarrow \text{Gumby} \\
DP &\rightarrow \text{Pokey} \\
VI &\rightarrow \text{wave} \\
VT &\rightarrow \text{ride} \\
C &\rightarrow \epsilon \\
I &\rightarrow \text{-s} \\
\nu P &\rightarrow \epsilon \\
\nu P2 &\rightarrow \epsilon
\end{aligned}
$$

The derivation tree for "ride -s Gumby Pokey" would be as follows, complete with derived string decorations:



B.4 INTERSECTION WITH AN FSM

The basic definition (not a serious algorithm; just a construction—see Seki *et al.* 1991 for a correctness proof) for intersection of an MCFG with an FSM is as follows:

Let $G = \langle N, \Sigma, F, P, S\rangle$ be an m-MCFG which generates L and $M = \langle Q, \Sigma, \delta, q_0, A\rangle$ be an FSM which accepts R. We construct an $m-$MCFG $G' = \langle N', \Sigma, F, P', S'\rangle$ that generates $L \cap R$ as follows:

1. $N' = \{S'\} \cup \{A[p_1, q_1, p_2, q_2, \ldots, p_{d(A)}, q_{d(A)}] | A \in N, p_i, q_i \in Q, 1 \leq i \leq d(A)\}$
2. P' is the smallest set of which the following requirements hold.

(a) For each rule

$$A_0 \to f[A_1, A_2, \ldots, A_{a(f)}] \in P$$

and

$$A'_i = A_i[p_1^{(i)}, q_1^{(i)}, \ldots, p_{d(A_i)}^{(i)}, q_{d(A_i)}^{(i)}], 0 \leq i \leq a(f)$$

which satisfy the *connecting condition* (see below), let

$$A'_0 \to f[A'_1, A'_2, \ldots, A'_{a(f)}] \in P'$$

.

(b) For all $q_F \in A$, $S' \to S[q_0, q_F] \in P'$.

B.4.1 CONNECTING CONDITION

B.4.1.1 *Nonterminating* f

Let each component f^h $(1 \leq h \leq r(f) = d(A_0))$ of f be

$$f^h[\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_{a(f)}] = z_{h1} z_{h2} \ldots z_{h\nu_h(f)},$$

where

$$z_{hk} = x_{i_{(h,k)} j_{(h,k)}}, 1 \leq k \leq \nu_h(f), 1 \leq i_{(h,k)} \leq a(f), 1 \leq j_{(h,k)} \leq d_{i_{(h,k)}}(f).$$

Then the following conditions must hold:

1. $p_{j_{(h,1)}}^{(i_{(h,1)})} = p_h^{(0)}$,
2. $p_{j_{(h,k)}}^{(i_{(h,k)})} = q_{j_{(h,k-1)}}^{(i_{(h,k-1)})}, 2 \leq k \leq \nu_h(f)$, and
3. $q_h^{(0)} = q_{j_{(h,\nu_h(f))}}^{(i_{(h,\nu_h(f))})}$.

B.4.1.2 *Terminating* f

Here the original rule in P is $A_0 \to \alpha$, $\alpha \in \Sigma^*$ and the corresponding intersection rule is $A_0[p_1^{(0)}, q_1^{(0)}] \to \alpha$. The connecting condition here is that

$$q_1^{(0)} \in \hat{\delta}(p_1^{(0)}, \alpha).$$

APPENDIX C

A Bottom-Up Algorithm for MCFG/WFSM Intersection

Intersection of MCFGs with WFSMs may be accomplished via a modified CYK chart parsing algorithm in the deductive chart parsing tradition (Shieber *et al.* 1995). I will introduce the full weighted finite state intersection algorithm in five steps: first, an overview of deductive chart parsing; second, a presentation of a bottom-up (CYK) chart parser for MCFGs; third, a sketch of a correctness proof; fourth, a modification of the parser to intersect MCFGs with FSMs instead of simply parsing sentences; fifth, a modification of the intersection algorithm to deal with weight minimization; and sixth, an overview of the implementation with some notes on complexity.

C.1 DEDUCTIVE CHART PARSING

A deductive chart parser attempts to construct the parse tree of a sentence under a particular grammar by using a special-purpose deductive system to prove that the sentence can be derived in the grammar. In order to avoid duplication of effort, the chart parser stores all formulas of the deductive system (known as *chart items*) as they are derived. A bottom-up deductive chart parser uses the words of the sentence—what will be the leaf nodes of the derivation tree—as the axioms of its deductive system and the productions

of the grammar as derivation rules. Each chart item makes some claim about licit subtrees built up from parts of the sentence being parsed. The deductive procedure used here is taken from Shieber *et al.* 1995; for a correctness proof see that article.

The procedure employs a chart, as mentioned above, in order to store items as they are computed, avoid redundant computation, and allow reconstruction of a parse three when the algorithm has completed. In addition the procedure makes use of an agenda to keep track of items to which the rules of deduction have not yet been applied. Overall, the procedure is as follows.

1. Initialize the chart and the agenda to contain the axiomatic items provided by the deductive system.

2. Repeat until the agenda is empty:

   (a) Select and remove an item from the agenda. This item will be referred to as the *trigger* item.

   (b) Generate all items that can be derived from the trigger item and zero or more items from the chart by one application of a rule of inference, and add these items to the chart if they are not already there. If an item is added to the chart, add it to the agenda as well.

3. If a goal item is in the chart, the sentence is recognized as a member of L(G), otherwise it is not.

C.2 BOTTOM-UP MCFG PARSING

The following deductive system is sufficient to parse an input string $w = w_1 w_2 \dots w_n$ against an MCFG $G = \langle N, \Sigma, F, P, S \rangle$ as defined in §B.1.1. The items of the deductive system are of the form $A[p_1, q_1, p_2, q_2, \dots, p_{d(A)}, q_{d(A)}]$ for $0 \leq p_i \leq q_i \leq n, 1 \leq i \leq d(A)$, $A \in N$. Such an item constitutes an assertion of existence of a derivation tree τ for grammar G with the following properties:

1. The head of τ is labeled by A.
2. The yield of τ, that is, the string for which τ is a derivation tree, is the tuple

$$\theta = \langle w_{q_{10}+1} \dots w_{q_{11}}, w_{q_{20}+1} \dots w_{q_{21}}, \dots, w_{q_{d(A)0}+1} \dots w_{q_{d(A)1}} \rangle.$$

   By the definition of a derivation tree, $\theta \in L_G(A)$.

C.2.1 AXIOMS

For each terminating rule $A \to \alpha$ in P such that $\alpha = w_{p+1} \dots w_q$ $(0 \leq p \leq q \leq n)$[1], there will be an axiom $A[p, q]$ in the deductive system.

C.2.2 GOAL

The goal item which indicates a successful parse is $S[0, n]$. This is because the interpretation of this item is that a derivation tree headed by S exists that yields the string $w_1 \dots w_n$, which is exactly what the parser is attempting to show.

[1] If $i = j$ then $\alpha = \epsilon$.

C.2.3 RULE OF INFERENCE

Unlike many deductive chart parsers, this one contains only a single rule of inference, albeit a rather general one:

$$\frac{A_1[p_1^{(1)}, q_1^{(1)}, \ldots, p_{d(A_1)}^{(1)}, q_{d(A_1)}^{(1)}], \ldots, A_{a(f)}[p_1^{(a(f))}, q_1^{(a(f))}, \ldots, p_{d(A_{a(f)})}^{(a(f))}, q_{d(A_{a(f)})}^{(a(f))}]}{A_0[f((p_1^{(1)}, q_1^{(1)}, \ldots, p_{d(A_1)}^{(1)}, q_{d(A_1)}^{(1)}), \ldots, (p_1^{(a(f))}, q_1^{(a(f))}, \ldots, p_{d(A_{a(f)})}^{(a(f))}, q_{d(A_{a(f)})}^{(a(f))}))]}$$

if $A_0 \rightarrow f[A_1, A_2, \ldots, A_{a(f)}] \in P$ and

$$f((p_1^{(1)}, q_1^{(1)}, \ldots, p_{d(A_1)}^{(1)}, q_{d(A_1)}^{(1)}), \ldots, (p_1^{(a(f))}, q_1^{(a(f))}, \ldots, p_{d(A_{a(f)})}^{(a(f))}, q_{d(A_{a(f)})}^{(a(f))}))$$

obeys the connecting condition of §B.4.1.1.

C.3 CORRECTNESS PROOF SKETCH

To prove this deductive system correct, it is necessary to prove soundness (that derivation of the goal item implies that the string is a member of the language) and completeness (that for any string in the language a goal item will be derived).

C.3.1 SOUNDNESS

This amounts to showing that the axioms are sound (that is they are only added if the claim they make is correct) and that the inference rules are sound as well (from correct antecedents they derive only correct consequences).

C.3.1.1 *Axioms*

The axiom $A : [p, q]$ asserts the existence of a tree $\tau \in CL(G)$ with the following properties:

1. The head of τ is labeled A.
2. τ has one child, the leaf labeled $\gamma = w_{p+1} \dots w_q$
3. The yield of τ is $\gamma = w_{p+1} \dots w_q$.

This claim is transparently correct by the definition (part 1) of $CL(G)$ and the definition of the axioms.

C.3.1.2 *Rule*

Since the rule of inference follows exactly the definition of a derivation tree, its application from derivation trees will produce derivation trees.

C.3.1.3 *Goal*

If the goal has been derived then by the definition of an item the sentence w is derived from S and therefore $w \in L(G)$.

C.3.2 COMPLETENESS

This amounts to showing that given a sentence $w = w_1 \dots w_n$ in the language, any derivation tree of that sentence will be derived by the system.

C.3.2.1 *Lemmas*

First, note that by the definitions of membership in the language and of derivation trees, any sentence in the language will have one or more associated derivation trees. Pick an arbitrary one, call it τ. τ has the following properties, from the definition of a derivation tree:

1. Each leaf node of τ is labeled by a nonterminal symbol paired with a contiguous subsequence $w_{p+1} \dots w_q$ for some p, q such that $0 \leq p \leq q \leq n$. The subsequence is contiguous because by the definition of MCFG all $f \in F$ have only the power to concatenate strings, not to rearrange the parts of strings or pull out subparts.

2. The string labels of the leaves of the tree can be placed in some linear sequence such that the ordered concatenation is equal to w. This linear sequence is furthermore such that each leaf label appears exactly once in it.

These properties in fact extend to any cut through the derivation tree, modified only by the fact that interior nodes can be labeled by a tuple of subsequences

$$\langle w_{p_1+1} \dots w_{q_1}, w_{p_2+1} \dots w_{q_2}, \dots, w_{p_m+1} \dots w_{q_m} \rangle,$$

for some m.

The proof that the item $S : [0, n]$ will be derived for any sentence $w = w_1 \dots w_n$ if τ, a derivation tree for w, is in $CL(G)$, is by induction, showing that, first, the leaves of τ are represented in the chart, and second, that if all τ nodes of maximum distance k from a leaf node in τ are represented in the chart, then all τ nodes of maximum distance $k + 1$ are represented as well.

C.3.2.2 *Basis Step*

If a leaf node $A : \gamma$ is in τ, then

1. There must be a production $A \rightarrow \gamma$ in the grammar, and
2. γ must be a subsequence $w_{p+1} \ldots w_q (0 \leq p \leq q \leq n)$ of w (by the lemmas).

This is the exact condition under which an axiom $A : [p, q]$ will be entered into the chart. Therefore the nodes of distance 0 are represented, since $A : \gamma$ was arbitrary.

C.3.2.3 *Inductive Step*

Assume that all τ nodes of maximum distance k from a leaf node are in the chart. Then take an arbitrary node $A : \theta$ in τ of maximum distance $k+1$ from a leaf node. By the definition of derivation tree, some production $A \rightarrow f[A_1, A_2, \ldots, A_{a(f)}]$ must exist in P such that the node $A : \theta$ has $a(f)$ children where child i is labeled $A_1 : \theta_1$ and $f[\theta_1, \theta_2, \ldots, \theta_{a(f)}] = \theta$. Furthermore the connectivity condition must hold, since parts 1 and 3 of the connectivity condition are just consequences of the way f is defined, and condition 2 is a consequence of the fact that θ cannot be part of a successful derivation if it contains subsequences $w_{p_1+1} \ldots w_{q_1} w_{p_2+1} \ldots w_{q_2}$ where $q_1 \neq p_2$, since no $f \in F$ has the power to fill in the middle of a string, only to concatenate its arguments in an order of its own determining. Therefore the connectivity condition is met and the chart contains the antecedents for the rule of derivation that will add the item $A : [p_1, q_1, p_2, q_2, \ldots, p_{d(A)}, q_{d(A)}]$ where $\theta = \langle w_{p_1+1} \ldots w_{q_1}, w_{p_2+1} \ldots w_{q_2}, \ldots, w_{p_{d(A)}+1} \ldots w_{q_{d(A)}} \rangle$, corresponding exactly to $A : \theta$.

Since $A : \theta$ was arbitrary, all nodes at maximum distance $k + 1$ from some leaf node will be generated.

Since all nodes of the derivation tree τ are generated, the top node of which must be $S : w$ by the definition of a derivation tree, the algorithm recognizes any arbitrary sentence in the grammar, and thus the completeness property holds.

C.4 CHART PARSING FOR FSM INTERSECTION

It should be clear from examining the items used by the previous algorithm and the definition of well-definition that the algorithm relates quite closely to the FSM intersection construction of §B.4. The primary difference is that the algorithm from the previous section has as its input a string of lexical items rather than a finite state machine over those items. To modify this into an FSM intersection algorithm requires only a few changes: items will be interpreted in terms of the FSM, axioms will be derived in a slightly different way, the goals will differ slightly, and a grammar recovery step will need to be added to retrieve the intersection grammar from the chart (a nearly equivalent algorithm can be used with sentence parsing to recover the parse tree). The sections to follow describe the algorithm for intersection of an FSM $M = \langle Q, \Sigma, \delta, q_0, A \rangle$ with an MCFG $G = \langle N, \Sigma, F, P, S \rangle$.

C.4.1 INTERPRETATION OF AN ITEM

An item $A[p_1, q_1, p_2, q_2, \ldots, p_{d(A)}, q_{d(A)}]$ indicates the existence of a derivation tree $\tau \in CL(G)$ with the following properties:

1. The root node of τ is labeled A.

2. The yield of τ is $L(M_1) \cap L_G^{(1)}(A) \times L(M_2) \cap L_G^{(2)}(A) \times \cdots \times L(M_{d(A)}) \cap L_G^{(d(A))}(A)$, where $M_i = \langle Q, \Sigma, \delta, p_i, \{q_i\}\rangle$ and $L_G^{(i)}(A) = \{\alpha_i | \langle \alpha_1, \alpha_2, \ldots, \alpha_i, \ldots, \alpha_{d(A)}\rangle \in L_G(A)\}$.

C.4.2 AXIOMS

For each arc $\langle p, \gamma, q\rangle \in \delta$ for which there is a terminating rule $A \to \gamma \in P$, add axiomatic item $A : [p, q]$.

C.4.3 GOALS

The goal items are $S : [q_0, q_F]$ such that $q_F \in A$.

C.4.4 GRAMMAR RECOVERY

The grammar recovery algorithm operates as follows:

1. Add all goal items in the chart to an agenda.

2. While the agenda is not empty:

   (a) Remove an item $A_0[p_1, q_1, p_2, q_2, \ldots p_{d(A_0)}, q_{d(A_0)}]$ from the agenda.

(b) For all rules $A_0 \rightarrow f[A_1, A_2, \ldots, A_{a(f)}] \in P$ look up all combinations of items headed by $A_1 \ldots A_{a(f)}$. For each combination, check whether f is well-defined. If so, add rule

$$A_0[p_1, q_1, p_2, q_2, \ldots p_{d(A_0)}, q_{d(A_0)}] \rightarrow f[A_1[p_1^{(1)}, q_1^{(1)}, \ldots, p_{d(A_!)}^{(1)}, q_{d(A_1)}^{(1)}], \ldots, A_{a(f)}[p_1^{(a(f))}, q_1^{(a(f))}, \ldots, p_{d(A_{a(f)})}^{(a(f))}, q_{d(A_{a(f)})}^{(a(f))}]]$$

to the intersection grammar. Add the item

$$A_i[p_1^{(i)}, q_1^{(i)}, \ldots, p_{d(A_i)}^{(i)}, q_{d(A_i)}^{(i)}]$$

to the agenda for every i if the item has not yet been added to the agenda.

Although step (2b) looks rather inefficient, it is possible to use the connectivity condition to speed item lookup.

The grammar produced by this process is transparently that produced in the intersection construction, so its language is $L(G) \cap L(M)$.

C.5 WEIGHTS

Modifying the MCFG/FSM intersection to introduce weights and weight minimization is a straightforward matter of adding a weight value to each item and adjusting to take the weight value into account. An item now takes the form $A[p_1, q_1, p_2, q_2, \ldots, p_{d(A)}, q_{d(A)}]/\omega$. The input to the algorithm is now a Weighted Finite State Machine (WFSM).

C.5.1 ITEM INTERPRETATION

Here I am assuming that weights come from the FSM only; weights from an MCFG could equally well be incorporated, but I cannot for the moment think of how that would be relevant to Optimality Theory. An item $A[p_1, q_1, p_2, q_2, \ldots, p_{d(A)}, q_{d(A)}]/\omega$ is interpreted as asserting the existence of a derivation tree $\tau \in CL(G)$ such that:

1. The root node of τ is labeled A.

2. The yield of τ is as defined in §C.4.1. Furthermore, there exists at least one $\theta = \langle \gamma_1, \gamma_2, \ldots, \gamma_{d(A)} \rangle$ in the yield of τ such that if $\omega_i(\gamma_i)$ is defined as $\min\{\omega' | (q_i, \omega') \in \hat{\delta}(p_i, \gamma_i)\}$ then $\Sigma_{i=1}^{d(A)} \omega_i(\gamma_i) = \omega$. In other words, at least one element of $Y(\tau)$ has weight ω.

C.5.2 AXIOMS

For each arc $\langle p, \gamma, q, \omega \rangle \in \delta$, if $A \to \gamma \in P$, add $A[p, q]/\omega$ to the agenda.

C.5.3 GOALS

The goal items are $S[q_0, q_F]/\omega$ such that $q_F \in A$. The ω parameter may carry any value.

C.5.4 RULE OF INFERENCE

The rule of inference is almost the same, except for the weights. The following rule abbreviates formulas which do not change as Γ (*e.g.*, $\Gamma_1 = A_1[p_1^{(1)}, q_1^{(1)}, \ldots, p_{d(A_1)}^{(1)}, q_{d(A_1)}^{(1)}]$)

and just gives the formula for weights:

$$\frac{\Gamma_1/\omega_1, \Gamma_2/\omega_2, \ldots, \Gamma_{a(f)}/\omega_{a(f)}}{\Gamma_0/\Sigma_{i=1}^{a(f)}\omega_i}$$

C.5.5 GRAMMAR RECONSTRUCTION

The weight-minimizing grammar recovery algorithm operates as follows:

1. Let I_g be the set of goal items in the chart, and let ω_{min} be $\min\{\omega | S[p, q]/\omega \in I_g\}$. Add the goal items $\{S[p, q]/\omega | \omega = \omega_{min}, S[p, q]/\omega \in I_g\}$ to the agenda.

2. While the agenda is not empty:

   (a) Remove an item $A_0[p_1, q_1, p_2, q_2, \ldots p_{d(A_0)}, q_{d(A_0)}]/\omega_0$ from the agenda.

   (b) For all rules $A_0 \rightarrow f[A_1, A_2, \ldots, A_{a(f)}] \in P$ look up all combinations of items headed by $A_1 \ldots A_{a(f)}$. For each combination, check whether f is well-defined and whether the weights sum correctly. If so, add rule

   $$A_0[p_1, q_1, p_2, q_2, \ldots, p_{d(A_0)}, q_{d(A_0)}] \rightarrow f[A_1[p_1^{(1)}, q_1^{(1)}, \ldots, p_{d(A_1)}^{(1)}, q_{d(A_1)}^{(1)}], \ldots, A_{a(f)}[p_1^{(a(f))}, q_1^{(a(f))}, \ldots, p_{d(A_{a(f)})}^{(a(f))}, q_{d(A_{a(f)})}^{(a(f))}]]$$

   to the intersection grammar. Add the item

   $$A_i[p_1^{(i)}, q_1^{(i)}, \ldots, p_{d(A_i)}^{(i)}, q_{d(A_i)}^{(i)}]/\omega_i$$

   to the agenda for every i if the item has not yet been added to the agenda.

That is, the algorithm is as before except that only minimum-weight goal items are used and weight sum values are checked.

C.5.6 NOTE

In the actual implementation it is not necessary to keep any but the lowest-weighted weight-variant of any item in the chart. That is, for any two items

$$A_0[p_1, q_1, \ldots, p_{d(A_0)}, q_{d(A_0)}]/\omega_1$$

and

$$A_0[p_1, q_1, \ldots, p_{d(A_0)}, q_{d(A_0)}]/\omega_2,$$

only item

$$A_0[p_1, q_1, \ldots, p_{d(A_0)}, q_{d(A_0)}]/\omega_{min},$$

where $\omega_{min} = \min\{\omega_1, \omega_2\}$, need be in the chart. If all weight-variants were allowed in the chart, it is possible, since constraints and candidate representations may be cyclic, that the chart would increase in size unboundedly. Thus, it is necessary that new items not be added to the chart that have higher weights than already present weight-variants.

C.6 IMPLEMENTATION NOTES

The implementation of the algorithms described here[2] relies on a slightly different normal form for MCFGs—it requires all nonterminating productions to have no more than two categories on the right hand side. An MCFG not in this normal form can be brought into it by means of the following transformation:

[2]Actually, two implementations exist at present: one for WFSM intersection written in C++ and one for MCFG parsing written in O'Caml. The C++ implementation is less general, having been optimized for the particular demands of Correspondence Theory Reduplication. Both are available at `http://www.humnet.ucla.edu/people/albro/software.html`.

While illegal productions exist:

1. Take a production $A_0 \to f[A_1, A_2, \ldots, A_n]$ such that $n > 2$ and remove it from the grammar.

2. Add a new production $B \to g[A_1, A_2]$ where $d(B) = d(A_1) + d(A_2)$ and

$$g^h[\overline{x}_1, \overline{x}_2] = \begin{cases} x_{1h}, & 1 \leq h \leq d(A_1) \\ x_{2(h-d(A_1))}, & d(A_1) < h \leq d(B) \end{cases}$$

3. Add production $A_0 \to f'[B, A_3, \ldots, A_n]$ to the grammar where f' is the result of substituting x_{i-1j} for x_{ij} in f for $3 \leq i \leq n, 1 \leq j \leq d(A_i)$ and also $x_{1(j+d(A_1))}$ for x_{2j} $(1 \leq j \leq d(A_2))$.

Since step 1 removes an illegal production, step 2 adds a legal one and step 3 adds one with a smaller number of arguments than the one removed, eventually the loop will terminate with a legal binary grammar.

I will not specify the implementation in exact detail—the source code is freely available; I will limit myself to a brief description. Here is how it works:

1. Start with an MCFG G.

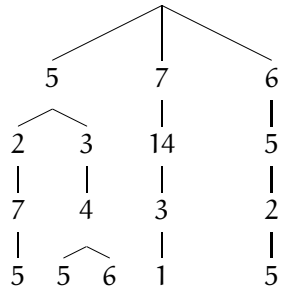
2. Analyze G and produce tables to speed up chart lookups based on the well-definition (connectivity) condition. For example, for a rule $A \to f[A_1, A_2]$ compute the dependencies that tell you given an item of type A_2 with particular p, q values what p, q values to use to look up a compatible A_1 item in order to form an item of type A. Also compute the dependencies for finding A_1, A_2 given A (used in grammar reconstruction).

3. Given an FSM to parse, produce axioms as described before.

4. Continue as described before, using the computed tables to speed chart lookup. If there are empty categories, assume their presence in the chart if possible rather than looking them up.

The chart was designed to be as compact and flexible as possible, at a possible expense of time complexity for some lookups. The chart will allow efficient lookup of any item or set of items where the category is known and the p, q values are specified as integers or wild-card values, *e.g.* $A[2, *, 5, 7]$ or $A[2, 5, *, *]$, which would both match, among other things, the item $A[2, 5, 5, 7]$.

The chart is represented as an array of sets of recursively nested tries. Each set of tries represents all of the items belonging to a particular category. For example, the trie



contains the items $A[5, 2, 7, 5]$, $A[5, 3, 4, 6]$, $A[7, 14, 3, 1]$, $A[6, 5, 2, 5]$, and $A[5, 3, 4, 5]$[3].

The estimated worst-case time complexity for parsing with a m-MCFG is no greater than $O(n^{3m})$ but should be less than that given limitations on p, q values. This is because there are $O(n^{2m})$ items max (less because of p, q limitations), and for each item it is necessary to look up no more than $O(n^m)$ items, where lookup is theoretically $O(1)$.

[3] Actually the order of the p, q values differs from the display order whenever so differing increases average lookup speed.

APPENDIX D

Relation to Gordon's (2002) Factorial Stress Typology

The constraints used here to analyze Malagasy stress are quite similar to those presented in Gordon (2002). In order that the origins and justification of the constraints used here may be more fully understood by reference with that work, I would like to present a mapping here from the terminology used there to that used here, although some constraints referenced there are inactive in Malagasy:

ALIGN(x_1, R, 0, PRWD) The effect of this constraint is to require that the left edge of a word should have a stress, and refers to a stress grid with levels 0, 1, and 2, where level 0 contains the syllables of the word (each syllable has a mark in the grid), 1 contains the secondary stress, and 2 contains primary stress(es). In my system this constraint is represented as *FINAL([V,-STRESS], C). There are slight differences between Gordon's constraint and mine—Gordon's constraint penalizes any stress which is not rightmost in the word, whereas mine penalizes any word which does not have a stress in its rightmost syllable. To make up this difference, my stress system requires the addition of lower-ranked *[+STRESS]. To make an exact equivalent of Gordon's constraint, you would have to consider tightly ranked *FINAL([V,-STRESS], C)$\gg$*[+STRESS] as a single constraint of sorts.

ALIGN(x_1, L, 0, PRWD) This is the mirror image of the previous constraint; its effect is to require that the left edge of a word should carry stress. The loose equivalent is *INITIAL([V,-STRESS], C), subject to the necessity for lower-ranked *[+STRESS] for full equivalence, as in the previous constraint.

ALIGN(x_2, R, 1, PRWD) This constraint penalizes any primary stress that is not the rightmost stress in a word. Its effect is to cause primary stress to be rightmost in a word. The loose equivalent for this constraint in my system is the pair *FINAL([+STRESS,-PRIM], [-stress]) (*RTMOST2NDARY) and *[+PRIM]. The system is only loosely equivalent however, as these two constraints allow only languages which require words to have primary stress or forbid them from having it. I believe that Gordon assumes the existence of a CULMINATIVITY constraint which would penalize words with no primary stress. The combination of CULMINATIVITY with ALIGN(x_2, R, 1, PrWd) should behave similarly to the combination of *RTMOST2NDARY and *[+PRIM].

ALIGN(x_2, L, 1, PRWD) This is the mirror image of the previous. The loose equivalent in my system is the constraint pair *INITIAL([+STRESS,-PRIM], [-STRESS]) and *[+PRIM].

ALIGN(EDGES, 0, PRWD, x_1) This constraint is violated once if a word has no left-aligned stress and also once if a word has no right-aligned stress (for a total of two possible violations per word). The equivalent in my system would be a constraint disjunction of *INITIAL([V,-STRESS], C) with *FINAL([V,-STRESS], C) (see Albro 1998b for the finite state formalization of constraint disjunction within the framework used here), although my guess would be that the separate existence of these two constraints is sufficient for all actual cases.

NonFinality: This constraint outputs a violation for any word in which the final syllable bears a stress. The equivalent here is *Final([V,+stress], C).

*Lapse: This constraint outputs a violation for any sequence of two adjacent unstressed syllables. The equivalent here is *Sequence([V,-stress], [V,-stress], C).

*ExtendedLapse: This constraint outputs a violation for any sequence of three adjacent unstressed syllables. The equivalent here is *DoubleSequence([V,-stress], [V,-stress], [V,-stress], C, C).

*Lapse-Right: Outputs a violation for any word in which the rightmost two syllables are unstressed. The equivalent here is *FinalSequence([V,-stress], [V,-stress], C).

*Lapse-Left: Outputs a violation for any word in which the leftmost two syllables are unstressed. The equivalent here is *InitialSequence([V,-stress], [V,-stress], C).

*ExtendedLapse-Right: Outputs a violation for any word in which the rightmost three syllables are unstressed. The equivalent here would be *FinalDoubleSequence([V,-stress], [V,-stress], [V,-stress], C, C).

*Clash: Outputs a violation for any sequence of two adjacent stressed syllables. The equivalent here is *Sequence([V,+stress], [V,+stress], C).

APPENDIX E

Diagrams of the Malagasy Analysis

The following are the final rankings and constraints used in the analysis. Note that some of these rankings were not mentioned in the text. For the most part, such unmentioned rankings are somewhat arbitrary choices required to ensure that randomly generated inputs respect the phonotactics of the language.

E.1 STEM LEVEL

The constraints used in the Stem level of the analysis divide into the strata shown in Table E.2. The necessary rankings that influence these strata are shown in Figure E.1.

E.2 WORD LEVEL

The constraints used in the Word level of the analysis divide into the strata shown in Table E.3, and necessary rankings that influence these strata are shown in Figure E.2.

DepIO	IdentIO(±[-nas,lab,C])	*t̲X
IO-Integrity(C)	IdentIO([±ant])	*Xʃ
MaxIB	IdentIO([+nas,lab])	*[-stress]/$_{red}$[C_0_
IdentIO([±syl])	IdentIO([dors])	*|[-prim][+prim]|
IdentIO([±nas])	*[+stress,-prim]/$_{RED}$[_]	*FinalLapse
IO-Uniformity	*Rtmost2ndary	WSP
IdentIO([V-place])	PreserveRt	IdentIO(Vs)/_C_0#
IdentIO([±vce])	Exists([+stress])	IdentIO(±[-nas,cor,C])
IdentIO([±strid])	*|[αplace,C][−αplace,C]|	IdentIO(±[+nas,cor])
IdentIO([±r])	*[αplace,C][−αplace,C]	
IdentIO([+nas,+x])	*|[αstress][-αstress]||	
*[+nas,+x]	IdentIO([±cont])	
IdentIO(±[cor,+nas,+x])	*[+prim]	
MaxIR(C)	*|[+cont][-cont]|	
IO-Integrity(V)		
NonFinality		
*Clash		
*Lapse		
IdentIO([+stress])		

Table E.2: Stem Level Strata

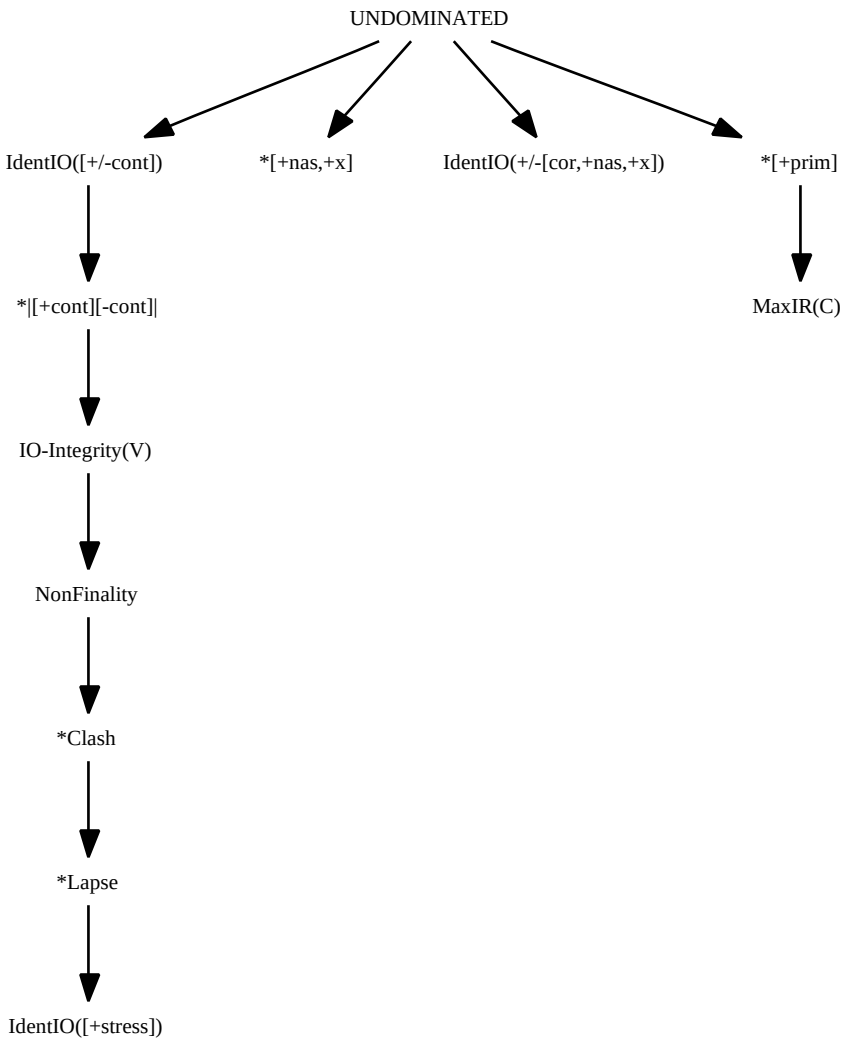


Figure E.1: Stem Level Hasse Diagram

DepIO	IO-Integrity	IdentIO([±nas])
IdentIO([±syl])	IO-Uniformity	IdentIO([cor])
IdentIO([+nas,+x])	MaxIO([+stress])	IdentIO([±round])
IdentIO([+high])	IdentIO([-low])	IdentIO([±back])
*Rtmost2ndary	*|[-prim][+prim]|	IdentIO([-high]) in [+stress]
IdentIO([+strid])	*u u	MaxIO(C)/_C
IdentIO([-vce])/_V	IdentIO([+vce])/N_	*[+nas,+x][-vce]
IdentIO([lab])/_V	*|[+cont][-cont]|	*[+nas,+x][+cont,-son,lab]
IdentIO([+ant])/_V	IdentIO([-high])/_V	*FinalLapse
*|[αstress][-αstress]|	OpqStress	IdentIO([+vce,+cont,-son]) in [+strid]
WSP	IdentIO([+stress])/_C_0#	IdentIO([lab]) in [+nas,+x]
*t̲ V	*[+cont,-son,dors]	IdentIO([+cont,-son,-ant,cor])
*X ʃ	*e a	*[+nas,dors],*[+nas,-ant,cor]
*t̲ #	IdentIO([-low])/_V	
*# ʃ	*i i	
MaxIO(C)/_V	*a V	
*[+nas,+x]	IdentIO([+nas,-ant,cor])	IdentIO([-strid,+fric,lab]) in β
*[-nas,+ant,cor,C]	IdentIO([+nas,dors])	MaxIO(V)/_V
*[lab,C]	*[-back,-high] C_0#	NonFinality
MaxIO	IdentIO([+cont,-son])/_V	
IdentIO([±vce])/#_	IdentIO([±lat])	*[+cont,-son]
IdentIO([-strid])	IdentIO([-stress,V])	IdentIO([-high])
IdentIO([+low])	*Clash	*[+prim]
IdentIO([lab])		
IdentIO([±cont])	*[-cor,+vce,+cont,-son]	*Lapse
IdentIO([+stress])		
IdentIO([+vce])/_V		
*[+vce,-son]		

Table E.3: Word Level Strata

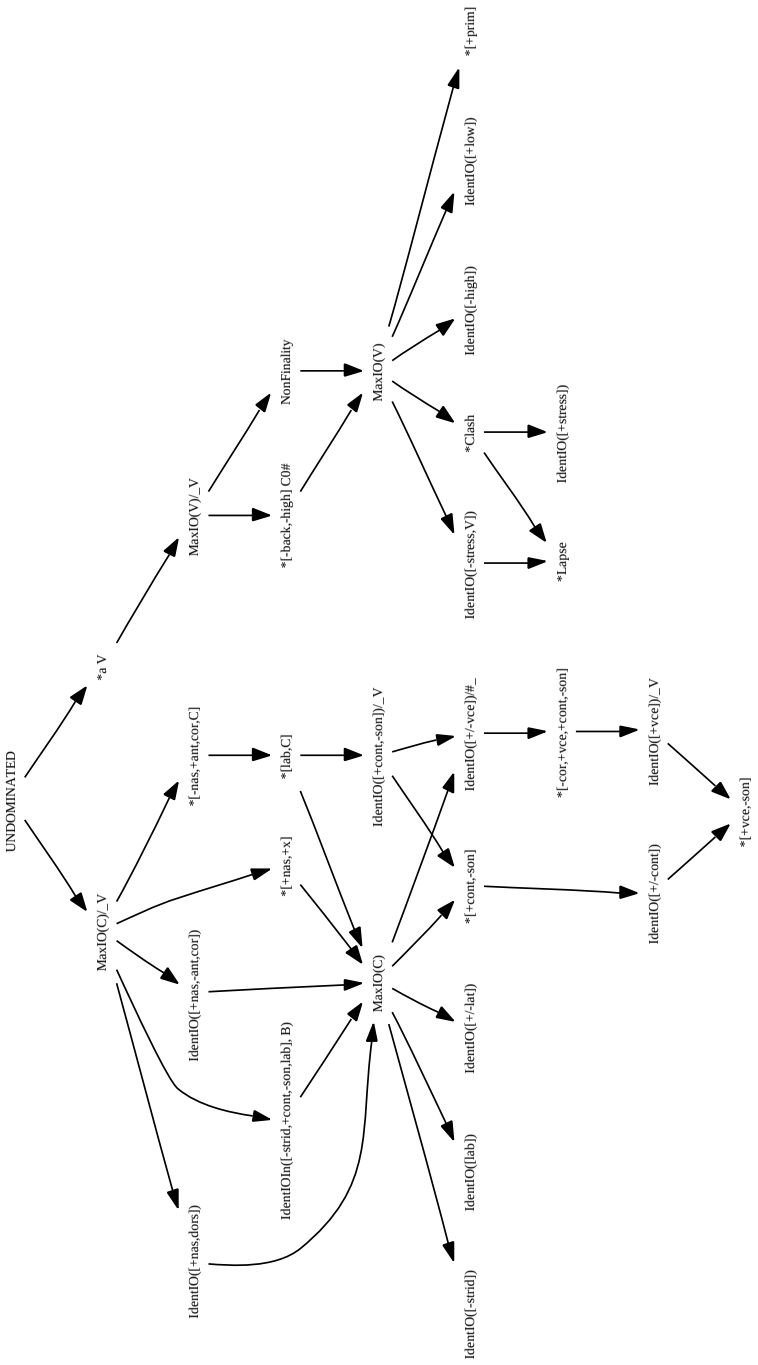


Figure E.2: Word-Level Hasse Diagram

E.3 POST-LEXICAL LEVEL

The post-lexical strata are shown in Table E.4 and the ranking diagram is Figure E.3.

DepIO(C)	IO-Integrity	IdentIO([±syl])
IdentIO([±nas])	IO-Uniformity(cont)	MaxIO([-nas,C])
MaxIO([+stress])	IdentIO([+stress])	*[-back,+low]
*[+nas,+x]	*[-strid,+cont,-son,lab]	IdentIO([±round])
*|ei|	IdentIO([+high])	*|[αnas][−αnas]|
IdentIO([-vce])/_V	*#[+nas][-vce]	*C [+nas]
*[-nas,C]#	Exists([+stress])	*|C V|
IdentIO([+ant])/_V	IdentIO([-ant,-nas,cor])/N_	IdentIO([+lab])/_V
*|[+son][-son]|	IdentIO([+vce])/_V	*|[αstress][-αstress]|
*[-cont][+cont,C]	*[αC-place][−αC-place]	*|[-prim][+prim]|
*|[+cont][-cont]|	*|[αC-place][−αC-place]|	*|[-cont][+cont,-cor]|
*Rtmost2ndary	IdentIO([-cor,-son])/_V	*|[αvce][−αvce]|
*|[-lat][+lat]|	*t̲ X	*X ʃ
*|[+high][-high]|	PtIdentIO([-son,-cont])	IdentIO([-stress])
*|[-low][+low]|		
IdentIO([±low])	DepIO(V)	*[+prim]
MaxIO(C)	IdentIO([-high])	*Clash
*N V#	*C C	IdentIO([±r])
IdentIO([-lab])		
IdentIO([cor])	IdentIO([±ant])	IO-Uniformity(C)
IdentIO([±vce])	IdentIO([+cont,-son])/_V	NonFinality
MaxIO(V)	FeatMaxIO([+cont,-son])/_V	
IdentIO([dors])	*[+cont,-son]	*Lapse
*[-low]		
FeatMaxIO([+cont,C])	IdentIO([±cont])	

Table E.4: Post-Lexical Level Strata

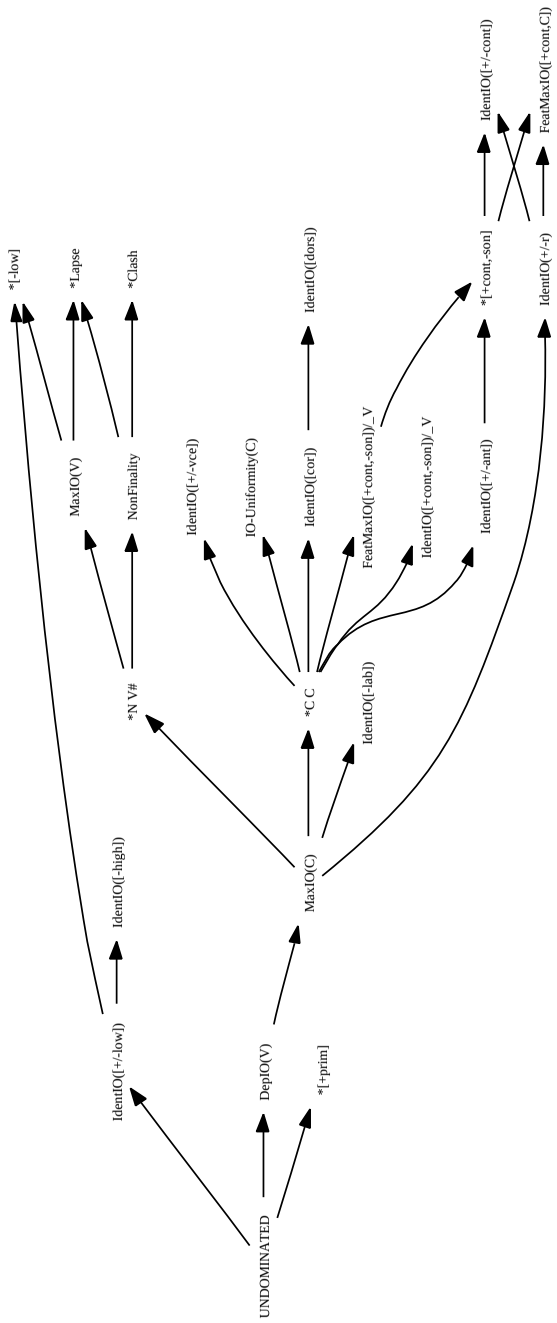


Figure E.3: Post-Lexical Level Hasse Diagram

APPENDIX F

A Finite-State Representation of Metathesis

The coïndexing representation of standard Correspondence Theory (McCarthy & Prince 1995) is overly powerful—an accurate implementation would require a potentially infinite random-access memory, thus such an implementation would necessarily be of greater complexity than a finite state machine or even a push-down automaton (equivalently, a context free grammar). In §3.2.2 I showed how to represent other forms of Input-Output correspondence with finite state methods; here is a representation for metathesis.

In this variant, the GEN function takes as its input an underlying form, as before, and produces a finite state machine representing an infinite set of candidates where each candidate is comprised of the underlying form paired with some output of the function *scramble* (to be described) and some surface form assembled from the list of valid segments. Such a machine may be constructed as follows:

> Let the underlying representation be represented by the symbol u and the ith segment of the underlying form by u_i. Further, let the output of the *scramble* function be a set S where each element of S is a reördering of the elements of u. Some cross-linguistic limit might be placed on the number of transpositions allowed in the output of *scramble*, so one might, for example,

define *scramble* as relating the underlying form to set of strings such that each string is identical to the underlying form except for having a single segment out of order. Let an arbitrary element of S be represented by the symbol s_j and the kth segment of s_j by s_{jk}.

The construction process then begins with machines M_{jk} (representing the kth underlying element in the candidate using scramble output s_j) where M_{jk} consists of an initial state q_1, a first middle state q_2, a second middle state q_3, and final state q_4. One arc for each permissible symbol (including the segments and also the segment divider symbol | and the non-correspondence symbol –) connects q_1 to q_2 (representing the surface form), a single arc labeled with segment u_k connects q_2 to q_3, and an arc labeled s_{jk} connects q_3 to q_4.

Now let $M_|$ be a finite state machine with states q_1, q_2, q_3, q_4, where all possible arcs connect q_1 to q_2 as in M_{jk}, and arcs labeled with the symbol | connect q_2 to q_3 and q_3 to q_4.

Next, define M_{jk}^{+} as the result of applying the Kleene plus operation to the machine M_{jk} and $M_{jk}^{+|}$ as the result of concatenating M_{jk}^{+} with $M_|$. The machine M_j is then defined as a concatenation beginning with $M_|$ and continuing with each $M_{jk}^{+|}$ for all k, in sequence. The final machine is the union machine for all M_j.

The above was a bit technical, but it should be clear from it that it is possible to construct a candidate set with candidates that look like the following example (underlying /asta/, surface [atsa]—coda simplification by local metathesis):

Output:	|	a	|	t	‖	s	|	a	|
Input:	|	a	|	s	|	t	|	a	|
Scrambled:	|	a	|	t	|	s	|	a	|

The usual Input-Output correspondence constraints of Correspondence Theory are then replaced by Scrambled-Output correspondence constraints, and IDENTIS (Input-Scrambled) constraints are used instead of the standard LINEARITY constraint. In this example all of the S-O constraints are satisfied, but IDENTIS([±CONT]) is violated in two places. A final example shows that non-local metathesis can be represented this way as well:

Output:	|	s	|	a	‖	t	|	
Input:	|	p	|	a	|	s	|	
Scrambled:	|	s	|	a	|	p	|	

The example has coda simplification by non-local metathesis as well as by change of articulator place, so both IDENTIS (for place and [±CONT]) and IDENTSO (for place only) are violated at various places.

References

Albro, Daniel M. 1994. AMAR: A computational model of autosegmental phonology. Bachelor's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

——, 1998a. Evaluation, implementation, and extension of Primitive Optimality Theory. Master's thesis, UCLA.

—— 1998b. Three formal extensions to Primitive Optimality Theory. In *Proceedings of the 17th International Conference on Computational Linguistics*.

—— 2000. A probabilistic ranking learner for phonotactics. In *Proceedings of the 2000 conference of the Linguistics Society of America*.

——, 2002. An Earley-style parser for Multiple Context-Free Grammars. At http://www.humnet.ucla.edu/people/albro/papers.html.

Beckman, J., 1998. *Positional Faithfulness*. University of Massachusetts dissertation.

Benua, Laura. 2000. *Phonological Relations Between Words*. Garland Publishing.

Bird, Steven, & T. Mark Ellison. 1994. One-level phonolgy: Autosegmental representations and rules as finite automata. *Computational Linguistics* 20.55–90.

Blevins, Juliette. 1997. Rules in Optimality Theory: Two case studies. In *Derivations and Constraints in Optimality Theory*, ed. by Iggy Roca, 227–260. New York: Oxford University Press.

Burzio, Luigi. 1995. Surface constraints versus underlying representations. In *Current Trends in Phonology: Models and Methods*, ed. by Jacques Durand & Bernard Laks. CNRS, Paris-X, and University of Salford Publications.

Chomsky, Noam A., & Morris Halle. 1968. *The Sound Pattern of English*. New York, NY: Harper and Row.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1.269–271.

Eisner, Jason, 1997a. Constraining OT: Primitive Optimality Theory. MIT Linguistics Lunch.

——. 1997b. Decomposing FootForm: Primitive constraints in OT. In *Proceedings of SCIL VIII*, MIT Working Papers in Linguistics.

——. 1997c. Efficient generation in Primitive Optimality Theory. In *Proceedings of the ACL*.

——, 1997d. What constraints should OT allow? Handout for talk at LSA, Chicago.

——. 2000. Directional constraint evaluation in Optimality Theory. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, 257–263, Saarbrücken, Germany.

Ellison, T. Mark. 1994a. The iterative learning of phonological rules. *Computational Linguistics* 20.

——. 1994b. Phonological derivation in Optimality Theory. In *Coling*, volume II, 1007–1013, Kyoto, Japan.

Erwin, Sean. 1995. Quantity and moras: An amicable separation. In *The Structure of Malagasy I*, ed. by Matthew Pearson & Ileana Paul, number 17 in UCLA Occasional Papers in Linguistics, 2–30. UCLA Linguistics Department.

Flemming, Edward, 1995. The analysis of contrast and comparative constraints in phonology. MIT Phonology Circle Lecture.

Frank, Robert, & Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics* 24.307–315.

Frisch, Stefan A., Janet B. Pierrehumbert, & Michael B. Broe. in press. Similarity avoidance and the OCP. *Natural Language and Linguistic Theory* .

Gazdar, Gerald. 1985. Applicability of indexed grammars to natural languages. Technical report, Center for Study of Language and Information.

Goldsmith, John A., 1976. *Autosegmental Phonology*. Massachusetts Institute of Technology dissertation.

Gordon, Matthew. 2002. A factorial typology of quantity insensitive stress. *Natural Language and Linguistic Theory* 20.491–552.

Harris, James W. 1969. *Spanish Phonology*. Cambridge, MA: MIT Press.

Hertz, Susan R. 1990. The Delta programming language: an integrated approach to nonlinear phonology, phonetics, and speech synthesis. In *Between the Grammar and Physics of Speech*, ed. by John Kingston & Mary E. Beckman, chapter 13, 215–257. New York, NY: Cambridge University Press.

Hollanger, Frederick S. 1973. *Diksionera/Malagasy-Englisy*. Lutheran Press and the American Cultural Center.

Hopcroft, John E. 1971. An algorithm for minimizing the states in a finite automaton. In *The theory of machines and computations*, ed. by Z. Kohavi, 189–196. New York: Academic Press.

——, & Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley.

Hume, Elizabeth, & Georgios Tserdanelis. 2003. Labial unmarkedness in Sri Lankan Portuguese Creole. *Phonology* 19.441–458.

Inkelas, Sharon, & Cheryl Zoll, 2000. Reduplication as morphological doubling. [Rutgers Optimality Archive #412].

Joshi, Aravind K., L. Levy, & M. Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences* 10.136–163.

Kager, René. 1999. *Optimality Theory*. Cambridge University Press.

Karttunen, Lauri. 1983. KIMMO: a general morphological processor. *Texas Linguistic Forum* 22.217–228.

——. 1998. The proper treatment of Optimality in Computational Phonology. In *Proceedings of the International Workshop on Finite-State Methods in Natural Language Processing*, 1–12, Bilkent University, Ankara, Turkey.

Keenan, Edward L., & Maria Polinsky. 1998. Malagasy (Austronesian). In *The Handbook of Morphology*, ed. by Andrew Spencer & Arnold M. Zwicky, chapter 28, 563–623. Blackwell.

——, & Jean Paulin Razafimamonjy. 1998. Reduplication in Malagasy. In *The Structure of Malagasy III*, UCLA Working Papers in Syntax and Semantics. UCLA Linguistics Department.

Kenstowicz, Michael. 1995. Base-identity and uniform exponence: Alternatives to cyclicity. In *Current Trends in Phonology: Models and Methods*, ed. by Jacques Durand & Bernard Laks. CNRS, Paris-X, and University of Salford Publications.

Kiparsky, Paul. 2000. Opacity and cyclicity. *The Linguistic Review* 17.351–367.

Kisseberth, Charles. 1970. On the functional unity of phonological rules. *Linguistic Inquiry* 1.291–306.

Koskenniemi, Kimmo. 1983. Two-level morphology: A general computational model for word-form recognition and production. Phd thesis, University of Helsinki, Helsinki, Finland.

McCarthy, John, & Alan Prince. 1993. Prosodic Morphology I: Constraint interaction and satisfaction. Technical Report 3, Rutgers University Center for Cognitive Science. [Rutgers Optimality Archive # 482].

——, & ——. 1995. Faithfulness and reduplicative identity. In *Papers in Optimality Theory*, ed. by J. Beckman, S. Urbanczyk, & L. Walsh, number 18 in University of Massachusetts Occasional Papers, 259–384. UMass, Amherst: GLSA.

McCarthy, John Joseph, 1995. Extensions of faithfulness: Rotuman revisited. Ms, University of Massachusetts, Amherst.

——, 1998. Sympathy and phonological opacity. Unpublished ms. [Rutgers Optimality Achive # 252].

McCawley, James D. 1974. Review of Chomsky and Halle, *The Sound Pattern of English*. *International Journal of American Linguistics* 40.50–88.

Pater, Joe. 1999. Austronesian nasal substitution and other NC effects. In *The Prosody-Morphology Interface*, ed. by René Kager, Harry van der Hulst, & Wim Zonneveld, chapter 8, 310–343. Cambridge University Press.

Paul, Ileana. 1995. The active marker and nasals in Malagasy. In *The Structure of Malagasy I*, ed. by Matthew Pearson & Ileana Paul, number 17 in UCLA Occasional Papers in Linguistics, 49–75. UCLA Linguistics Department.

Pearson, Matthew, 1994. Stress and vowel devoicing in Malagasy. UCLA ms.

Pollard, Carl J., 1984. *Generalized phrase structure grammars, head grammars, and natural language*. Stanford University dissertation.

Prince, Alan, & Paul Smolensky. 1993. Optimality Theory: Constraint interaction in generative grammar. Technical Report 2, Center for Cognitive Science, Rutgers University.

Pulleyblank, Douglas. 2003. When is a feature a feature? In *Proceedings of the 22nd West Coast Conference Conference on Formal Linguistics (WCCFL XXII)*.

Raimy, E. 2000. Remarks on backcopying. *Linguistic Inquiry* 31.541–552.

Ringen, Catherine O., & Orvokki Heinämäki. 1999. Variation in Finnish vowel harmony: An OT account. *Natural Language and Linguistic Theory* 17.303–337.

Seki, Hiroyuki, Takashi Masumura, Mamoru Fujii, & Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88.

Shieber, Stuart M., Yves Shabes, & Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming* 24.

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

Stabler, Edward P. 1997. Derivational minimalism. In *Logical Aspects of Computational Linguistics*, ed. by Christian Retoré, number 1328 in Lecture Notes in Computer Science, 68–95. NY: Springer-Verlag.

Steedman, Mark J. 1986. Combinators and grammars. In *Categorial Grammars and Natural Language Structures*, ed. by R. Oehrle, E. Bach, & D. Wheeler. Dordrecht: Foris.

—— 1987. Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory* .

Steriade, Donca. 1996. Paradigm uniformity and the phonetics-phonology boundary. In *5th Conference in Laboratory Phonology*, Evanston, Illinois.

——. 1998. Alternatives to the syllabic interpretation of consonantal phontactics. In *Proceedings of the 1998 Linguistics and Phonetics Conference*, ed. by O. Fujimura, B. Joseph, & B. Palek, 205–242. The Karolinum Press.

——. 2000 (in press). Directional asymmetries in place assimilation: a perceptual account. In *Perception in Phonology*, ed. by E. Hume & K. Johnson. Academic Press.

Tesar, Bruce. 1995. Computing optimal forms in optimality theory: Basic syllabification. Technical Report CU-CS-763-95, Department of Computer Science, University of Colorado, Boulder.

——. 1996. Computing optimal descriptions for optimality theory grammars with context-free position structures. In *Proceedings of ACL*.

Trigo, Lauren, 1988. *On the phonological deviation and behavior of nasal glides*. Massachusetts Institute of Technology dissertation.

Vijay-Shanker, K., David J. Weir, & Aravind K. Joshi. 1986. Tree adjoining and head wrapping. In *Proceedings of the 11th International Conference on Computational Linguistics*, 202–207.

——, ——, & ——. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th meeting of the Association for Computational Linguistics*, 104–111.

Walther, Markus. 2000. Finite-state reduplication in one-level prosodic morphology. In *Proceedings of NAACL-2000*, 296–302, Seattle, WA.

——. 2001. Correspondence theory: More candidates than atoms in the universe. Technical report, Institut für Germanistische Sprachwissenschaft Philipps-Universität Marburg. Archived as MAL-6, at http://www.uni-marburg.de/linguistik/mal.

Wilson, Colin, 2000. Targeted constraints, contextual neutralization and phonological opacity. UCLA Talk.