A Large-Scale, Computerized Phonological Analysis of Malagasy

Daniel M. Albro*

January 4, 2003

1 INTRODUCTION

Since the advent of Optimality Theory (Prince & Smolensky 1993) 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). Optimality Theory arguably has an advantage over rewrite rule phonology in terms of explanatory power, but its complexity makes it difficult to analyze a large body of data and phonological phenomena in a consistant and demonstrably correct way. The work described here uses computational methods to get around the difficulties posed by OT and points the way to a return of large-scale, whole-language analysis.

DATA COVERED The data covered by this analysis is the phonological pattern of the Merina dialect (the standard dialect) of Malagasy, the primary language of Madagascar. The analysis is based on a large and continually expanding database of Malagasy forms, derived from Hollanger's (1973) Malagasy dictionary, data from a native speaker, and several phonological papers (Erwin 1995; Keenan & Razafimamonjy 1995; Keenan & Razafimamonjy 1998; Paul 1995; Keenan & Polinsky 1998).

2 FRAMEWORK USED

The framework used in the analysis, as compared with others:

*albro@humnet.ucla.edu, http://www.humnet.ucla.edu/people/albro/albro.htm

output:	|	p	|	u	|	p	p	p	p	|	u	|	k	|	a	|
input:	|	p	|	u	|	k	k	|	p	|	u	|	k	|	–	–
RED-copy:	p	p	|	u	|	k	|	–	–	–	–	–	–	–	–	–	–
morphology:	B	B	B	B	B	B	B	R	R	R	R	R	R	R	–	–

Figure 1: Representation of [pupuka], derived from /puk+RED/.

2.1 REPRESENTATION

Candidates are represented as ASCII strings from an alphabet consisting of named segments, segment boundary symbols, a symbol for non-correspondence, and identificatory symbols for the reduplicant and the base. The representation was designed for computational reasons to be as simple as possible, while still carrying enough information for a complete analysis. It dispenses with syllables and feet, as these, while perhaps necessary for other languages, are not necessary to describe the phonology of Malagasy, which has an essentially CV syllable structure.

2.1.1 *Correspondence*

This analysis is implemented using finite state methods. These methods are not sufficiently powerful to be able to calculate constraint violations using the standard indexing model of correspondence. Instead the representation, following Primitive Optimality Theory (Eisner 1997; Albro 1998), is arranged in a sort of chart like a gestural score (see Figure 1), in which joint membership in a vertical time-slice (a column on the chart) indicates correspondence. If an element of some row has no correspondent in some other row, then that row will have the symbol "–" in the column to which the non-corresponding element belongs. Because reduplication involves correspondence between two parts of the same row, a special row called the *reduplicant reference level* is added to the chart. For each candidate, the reduplicant reference level contains an exact copy of the surface form of the reduplicant, placed into vertical alignment with the base. Another row contains morphological features—here, base, reduplicant, or neither.

The representation used here makes most of the correspondences representable via standard Correspondence Theory (McCarthy & Prince 1995) available, while still being implementable using finite state methods. It does not allow for crossing correspondences as in metathesis, although a finite-state representation for metathesis is

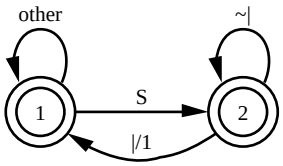

Figure 2: Schematic for the *(S) family of constraints

possible, albeit somewhat costly computationally[1].

2.2 IMPLEMENTATION AND GEN MODEL

The analysis is based on an implementation of Optimality Theory with an explicit model of GEN. This implementation is in the tradition of Ellison (1994), Eisner (1997), and Albro (1998), that is, of Optimality Theory implementations that use weighted finite state machines to represent constraints. Its major contribution to this tradition is its ability to generate reduplicated forms under a Correspondence Theory analysis. The implementation has the following characteristics:

- The candidate pool at any stage of the winnowing process is modeled by a finite state machine (FSM)[2].
- Constraints are weighted finite state machines (Figure 2 shows a schematic of a family of constraints, and Figure 3 shows a particular member of that family).
- Generation (leaving aside reduplication for the moment) is by the following algorithm, exemplified by a derivation of the reduplicated form [ˌbeˈbe]:

    1. Represent the candidate set as a finite state machine that specifies all ways in which the input form could correspond to a surface form. For our example, this set is depicted in Figure 4.

[1]This could be done by having two underlying forms—one in the surface order, and one in the underlying order. Differences between the two underlying forms would be construed as linearity violations.

[2]A device which can check whether an arbitrary string is a member of a specified set while using a finitely bounded amount of memory.

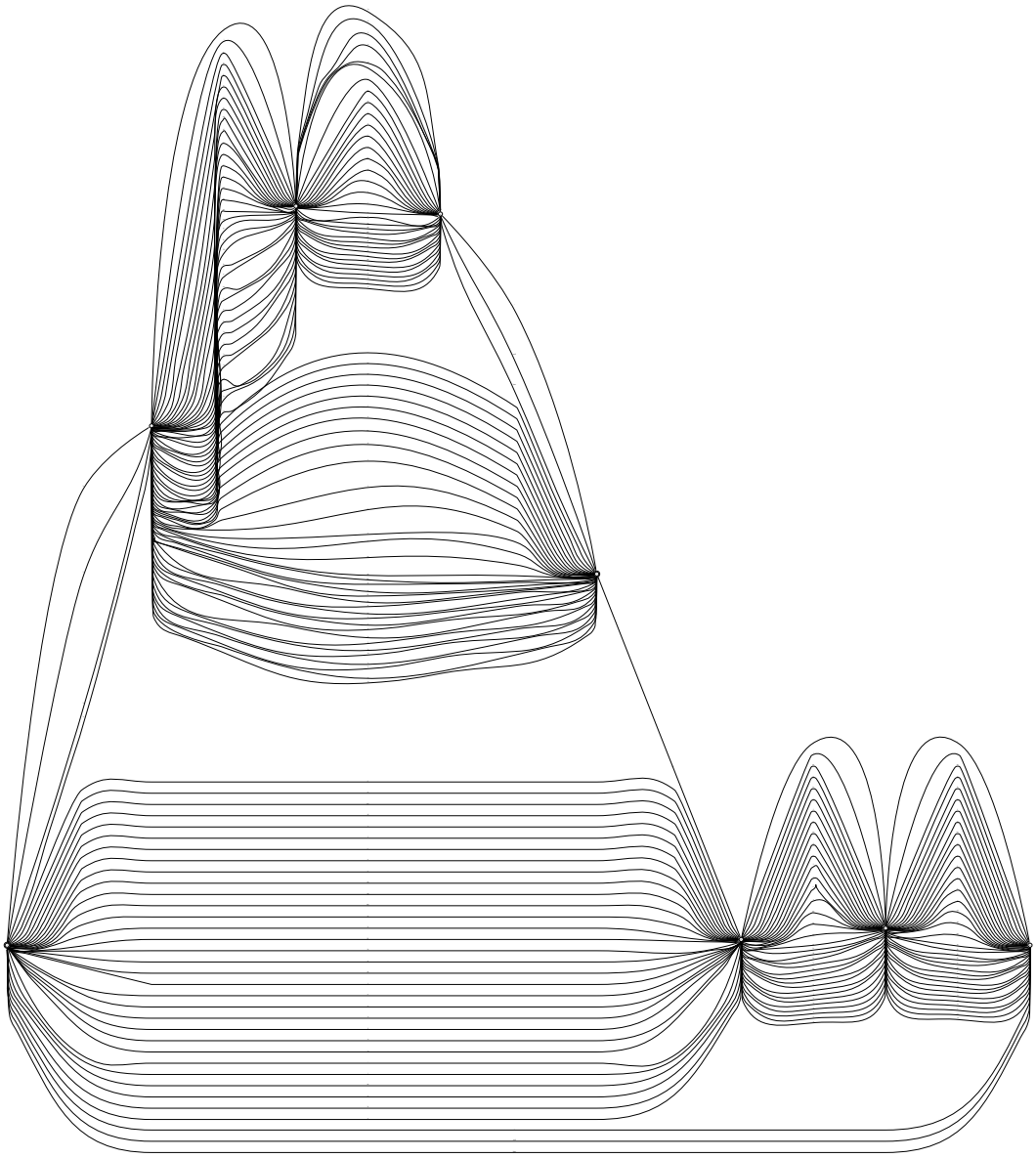

Figure 3: Representation of *(VPS).

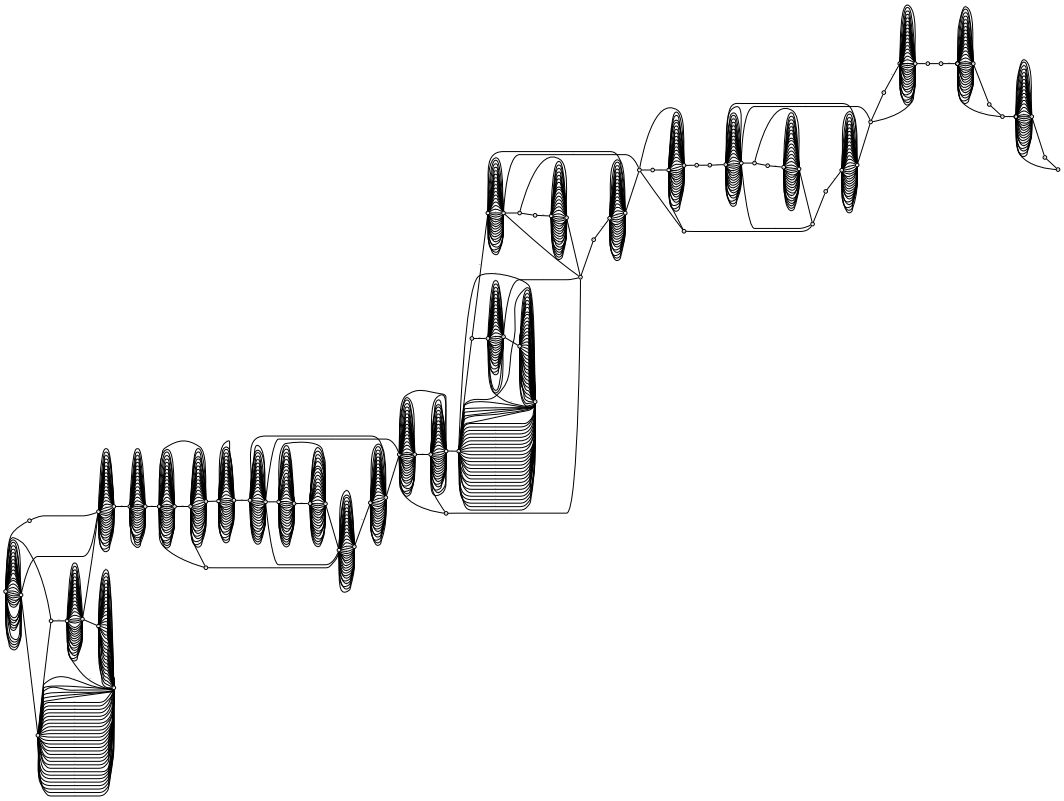

Figure 4: /be+RED/, with no output specified.

2. Intersect the candidate set with a constraint, which is a weighted finite state machine. The result is the same set of strings as in the candidate set, but with weights on the less-optimal candidates.
3. Apply a modified form of Dijkstra's (1959) Single-Source Shortest Paths algorithm (see Albro 1998 for a description of the modified algorithm) to prune non-optimal candidates. For our example the output of this stage is shown in Figure 5.
4. Go to (2) for the next lower ranked constraint until there are no more constraints or a base-reduplicant correspondence constraint is encountered. In the case where there are no more constraints, the candidate set contains the chosen output or outputs. For our example the final candidate set before the first base-reduplicant correspondence constraint is shown in Figure 6.

- Before the first Base-Reduplicant correspondence constraint can be intersected with the candidate set, each candidate must be such that its reduplicant reference level contains an exact copy of the surface reduplicant for that candidate so that the Base-Reduplicant correspondence relation can be computed. This is a long-distance copying dependency, and any grammar that can express such a dependency must fall significantly higher in the Chomsky hierarchy than a finite state machine. In particular, it must be at least as powerful as a mildly context sensitive grammar. The grammar formalism I have chosen here is the Multiple Context Free Grammar[3] (MCFG). Thus, the algorithm continues as follows:

  1. Use a chart parsing algorithm to intersect the candidate set FSM with an MCFG representing an identity requirement between the reduplicant and a copy of the reduplicant used as a reference for correspondence. The result is an MCFG representing the same set of surface strings, but where each candidate has an exact copy of the surface reduplicant placed in vertical correspondence with the base. For our example this MCFG is shown in Figure 7.

[3]A mildly context-sensitive grammar formalism. See Seki *et al.* 1991. They have generative power strictly between tree adjoining grammars or head grammars and context sensitive grammars. They are equivalent in generative power to linear context-free rewriting systems and Stabler's (1997) formalization of minimalist grammars (Chomsky 1995).

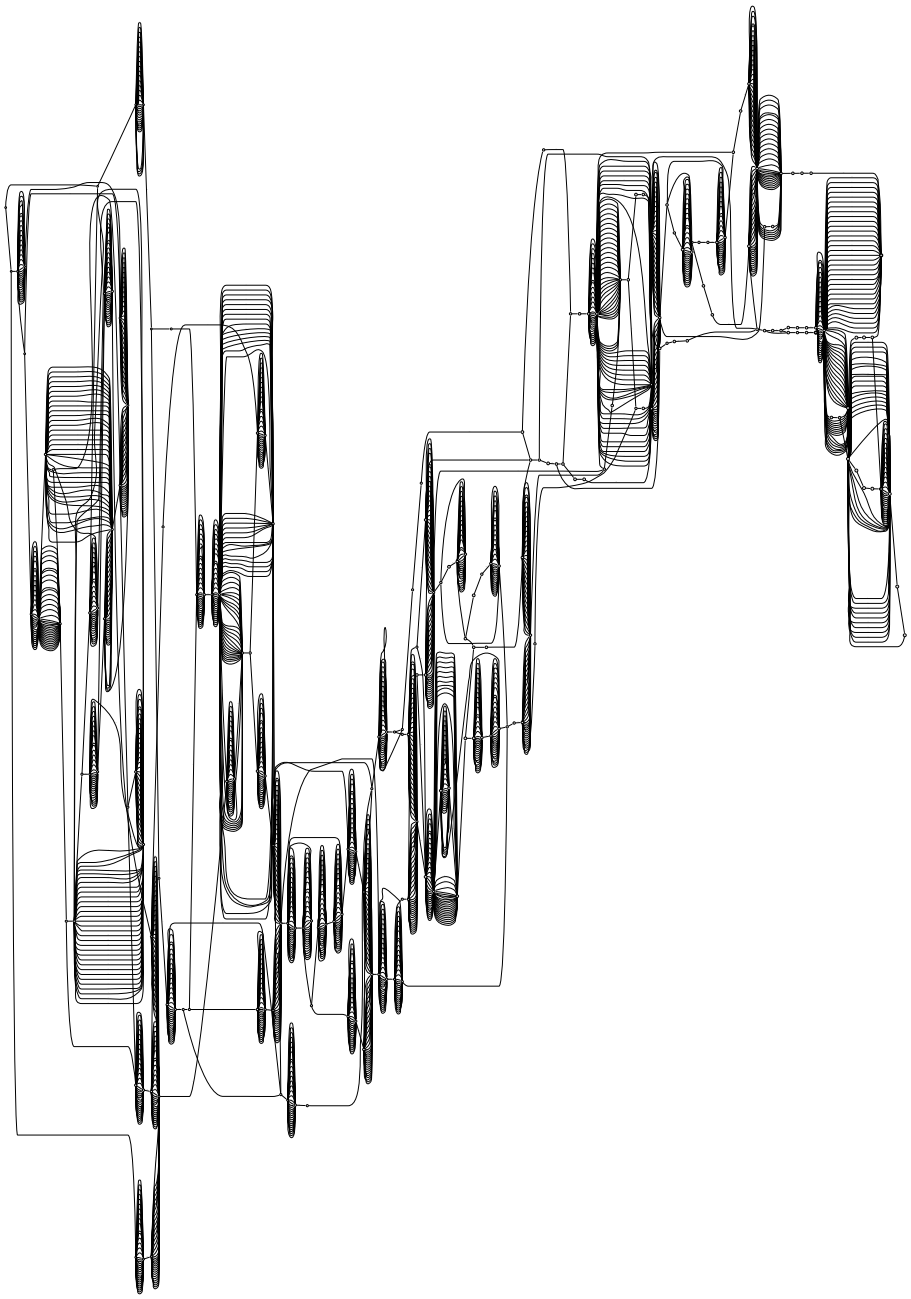

Figure 5: /be+RED/, after a few constraints.

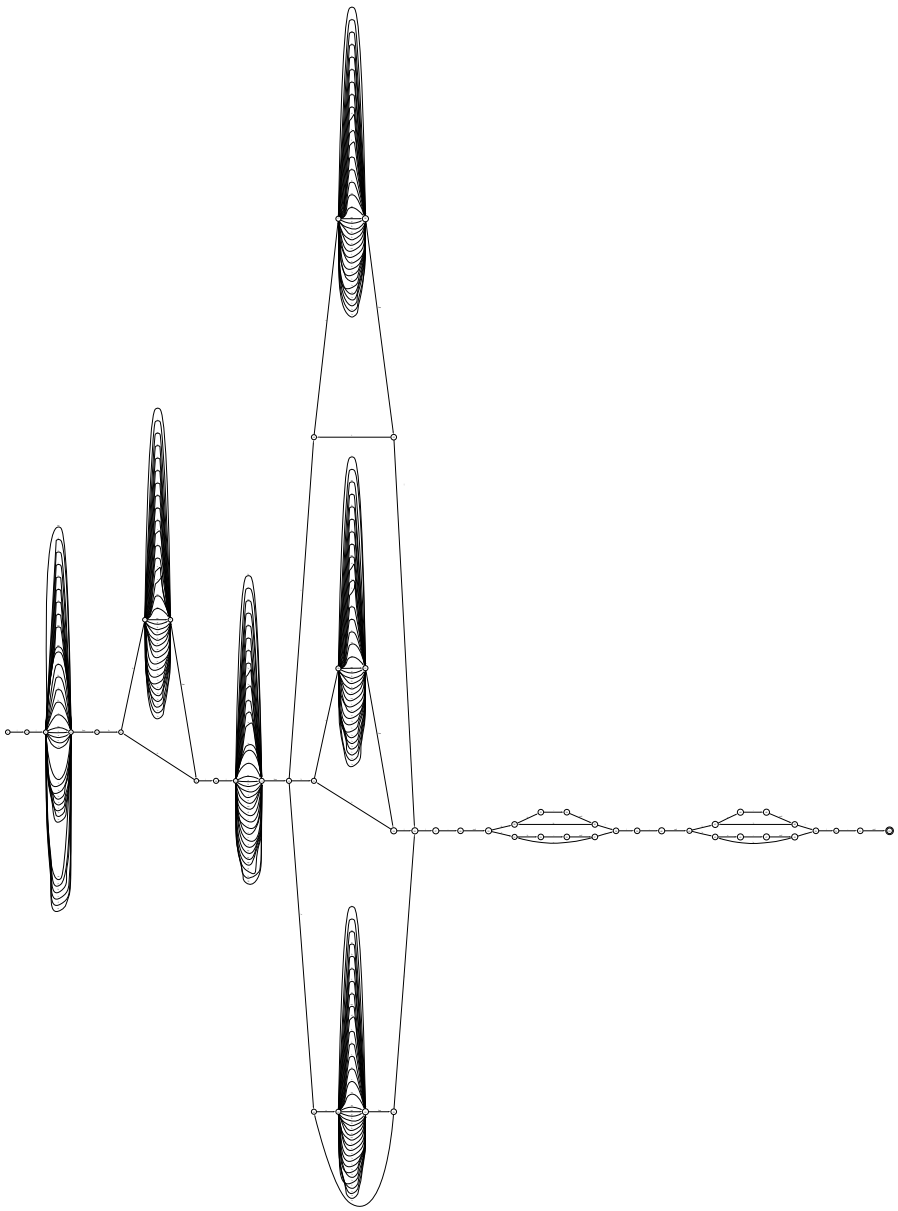

Figure 6: /be+RED/, just before first B-R correspondence constraint.

```
S-1 --> g1[Rd-2]
S-1 --> g1[Rd-3]
S-1 --> g1[Rd-4]
Rd-2 --> g3[Rda-5 Rd-6]
Rd-3 --> g3[Rda-5 Rd-7]
Rd-4 --> g3[Rda-5 Rd-8]
Rd-6 --> g3[Rda-11 Rd-12]
Rd-6 --> g3[Rda-13 Rd-14]
Rd-7 --> g3[Rda-11 Rd-15]
Rd-7 --> g3[Rda-13 Rd-16]
Rd-8 --> g3[Rda-11 Rd-17]
Rd-8 --> g3[Rda-13 Rd-18]
Rd-12 --> g3[Rda-11 Rd-12]
Rd-12 --> g3[Rda-22 Rd-23]
Rd-12 --> g3[Rda-24 Rd-25]
Rd-12 --> g3[Rda-5 Rd-26]
Rd-14 --> g3[Rda-13 Rd-14]
Rd-14 --> g3[Rda-22 Rd-23]
Rd-14 --> g3[Rda-28 Rd-29]
Rd-14 --> g3[Rda-5 Rd-26]
Rd-15 --> g3[Rda-11 Rd-15]
Rd-15 --> g3[Rda-22 Rd-30]
Rd-15 --> g3[Rda-24 Rd-31]
Rd-15 --> g3[Rda-5 Rd-32]
Rd-16 --> g3[Rda-13 Rd-16]
Rd-16 --> g3[Rda-22 Rd-30]
Rd-16 --> g3[Rda-28 Rd-33]
Rd-16 --> g3[Rda-5 Rd-32]
Rd-17 --> g3[Rda-11 Rd-17]
Rd-17 --> g3[Rda-22 Rd-34]
Rd-17 --> g3[Rda-24 Rd-35]
Rd-17 --> g3[Rda-5 Rd-36]
Rd-18 --> g3[Rda-13 Rd-18]
Rd-18 --> g3[Rda-22 Rd-34]
Rd-18 --> g3[Rda-28 Rd-37]
Rd-18 --> g3[Rda-5 Rd-36]
```

⋮

Figure 7: Initial MCFG for /be+RED/.

2. Intersect the candidate set with a constraint weighted finite state machine using weighted MCFG chart parsing (see Albro to appear for the algorithm). The result is the set of candidates which best satisfy the constraint. Repeat with the next constraint, until no more are left.
3. It is possible to recover the winning candidate set, then, by converting the final MCFG (shown for our example in Figure 8) into a loosely equivalent FSM. This FSM can then be modified to leave only the winning surface form or forms. The finite state representation of [ˌbeˈbe] thus produced is shown in Figures 9 and 10.

2.3 THE CONSTRAINT COMPONENT

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 vertical alignment definition of correspondence rather than the standard indexing method.
2. 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. These two-level constraints can be used to account for opacity, among other things.
3. They are uniformly implementable as weighted finite state machines, which rules out overly powerful families such as gradient CONTIGUITY or ALIGN, for example.

A definition of each of the families available in the implementation appears in Appendix B.

2.4 REDUPLICATION MECHANISM

REDUPLICANT SHAPE McCarthy & Prince (1995) control the shape of the reduplicant via generalized alignment constraints (the reduplicant edges must be the edges of a foot, a syllable, etc.) or other surface well-formedness constraints (but see McCarthy 2001 for a disavowal of generalized alignment).

```
S-1 --> g1[Rd-2]
Rd-2 --> g3[Rda-3 Rd-4]
Rda-3 --> g3[Rdb-5 BR-6]
Rd-4 --> g3[Rda-7 Rd-8]
Rdb-5 --> g8[B-9 O-10]
BR-6 --> bas:3 red:3
Rda-7 --> g3[Rdb-11 BR-6]
Rd-8 --> g3[Rda-3 Rd-12]
Rd-8 --> g3[Rda-7 Rd-8]
B-9 --> |:2 |:0
O-10 --> g9[Oa-13 Ob-14]
Rdb-11 --> g8[B-15 O-16]
Rd-12 --> g3[Rda-17 Rd-18]
Oa-13 --> g4[A-19 U-20]
Ob-14 --> g4[U-20 M-21]
B-15 --> b:2 b:0
O-16 --> g9[Oa-22 Ob-23]
Rda-17 --> g3[Rdb-24 BR-6]
Rd-18 --> g2[Rda-25]
Rd-18 --> g3[Rda-17 Rd-18]
A-19 --> |:0
U-20 --> |:1
M-21 --> -:2
Oa-22 --> g4[A-26 U-27]
Ob-23 --> g4[U-27 M-21]
Rdb-24 --> g8[B-28 O-29]
Rda-25 --> g3[Rdb-30 BR-6]
A-26 --> b:0
U-27 --> b:1
B-28 --> 'e:2 'e:0
O-29 --> g9[Oa-31 Ob-32]
Rdb-30 --> g8[B-9 O-33]
Oa-31 --> g4[A-34 U-35]
Ob-32 --> g4[U-35 M-21]
O-33 --> g9[Oa-31 Ob-14]
A-34 --> ‘e:0
U-35 --> e:1
```

Figure 8: Final MCFG for /be+RED/

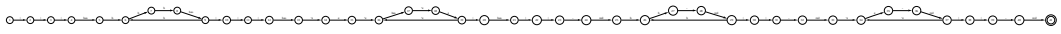

Figure 9: Final output for /be+RED/, as an FSM

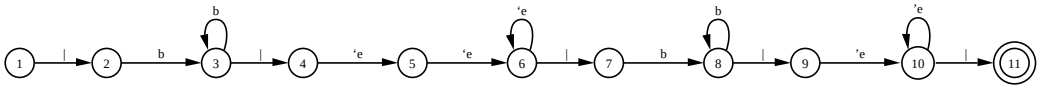

Figure 10: Final surface-only output for /be+RED/, as an FSM

Generalized alignment is too computationally powerful for a full finite-state implementation, but the same basic method is used here. For Malagasy, the requirement is that the leftmost syllable of the reduplicant must have primary stress, and that this must be the last stressed syllable in the word.

REDUPLICANT LOCATION McCarthy & Prince (1995) use the constraints ANCHORLFT and/or ANCHORRT to control the side of the base which is copied. The ANCHOR constraint says that the leftmost (rightmost) elements of the base and reduplicant must be in correspondence. This would go along with an alignment constraint that indicates where the reduplicant falls with respect to the base, and CONTIGUITY, which eliminates candidates in which non-contiguous parts of the base are preserved.

The analysis presented here cannot use this method, however. The gradient CONTIGUITY constraint described in McCarthy & Prince 1995 cannot be implemented without either going beyond the power of a weighted finite state machine or adding extra layers to the representation. Thus the analysis uses a constraint PRESERVERT/LFT which *is* implementable as a finite state machine and usurps the functions of both CONTIGUITY and ANCHOR. PRESERVERT outputs a violation for each MAXBR violation after the first non-violation of MAXBR within the reduplicant, and similarly PRESERVELFT outputs a violation for each MAXBR violation before the rightmost non-violation of MAXBR (See Table 1) . Since PRESERVELFT/RT can be satisfied by an empty reduplicant, the analysis also uses a simple existential faithfulness constraint to ensure that the reduplicant is not empty (this constraint, EXTMAXIR, is violated if there is any segment in the underlying reduplicant but no segments in the surface reduplicant—note that EXTMAXBR would work just as well for this) (*cf.* Kurisu's (2001) REALIZEMORPH constraint).

SR	h	a	d	i	n	u	h	ai	–	–	–	–	
UR	h	a	d	i	n	u	h	a	d	i	n	u	
RR	h	ai	–	–	–	–	–	–	–	–	–	–	
Morph	B	B	B	B	B	B	R	R	R	R	R	R	
Violations									*	*	*	*	
SR	h	a	d	i	n	u	–	–	d	i	n	–	
UR	h	a	d	i	n	u	h	a	d	i	n	u	
RR	–	–	d	i	n	–	–	–	–	–	–	–	☜
Morph	B	B	B	B	B	B	R	R	R	R	R	R	
Violations												*	
SR	h	a	d	i	n	u	h	a	–	–	n	–	
UR	h	a	d	i	n	u	h	a	d	i	n	u	
RR	h	a	–	–	n	–	–	–	–	–	–	–	
Morph	B	B	B	B	B	B	R	R	R	R	R	R	
Violations									*	*		*	

Table 1: PRESERVERT violations in losing and winning candidates for /hadinu+RED/

2.5 RANKING PECULIARITIES

Because chart parsing (or any other method that would allow for enforcement of long-distance identity relations in candidate strings) is more expensive computationally than finite state machine intersection, it is desirable to put it off until the candidate set has been reduced to a manageable level. For this reason, contrary to McCarthy & Prince (1995), the constraint ranking used for the Malagasy analysis was designed such that Input-Reduplicant constraints outrank Base-Reduplicant constraints wherever possible, and in general Base-Reduplicant constraints are ranked as low in the grammar as possible (or at least the highest-ranked Base-Reduplicant correspondence constraint is ranked as low as possible). Sometimes this was achieved by splitting constraints that would naturally be Input-Output correspondence constraints into Input-Base and Input-Reduplicant versions, where the Input-Base version is assumed to apply anywhere except inside a reduplicant. General well-formedness constraints are also split into specialized versions that apply either inside or outside of the reduplicant.

3 THE ANALYSIS

The actual analysis is too complex to explicate fully in this forum, although the ranking Hasse diagram is given in the appendix.

TOOLKIT The analysis was produced by OTPad, my toolkit for constructing large-scale analyses in Optimality Theory. The toolkit has the following features:

1. Graphical display and entry of phonological data, grouped into paradigms. (Figure 11)
2. Generation of outputs from inputs, with automatic detection of incorrect or ambiguous outputs. (Figure 12)
3. Automatic storage of incorrect outputs as candidates, and automatic driving of the generation algorithm with random constraint ranking variations to generate candidates. (Figure 13)
4. Automatic collection and summary of ranking information (arguments, diagrams, strata, etc.) given stored candidates.
5. Output and generation of tableaux in HTML, ASCII, and LaTeX formats. (Figure 14)
6. Automatic constraint ranking given expected outputs (no need for candidates). For example, given a grammar that accounts for a particular set of forms, you might add a new form that the current ranking does not properly account for. You could then specify the expected output and run this algorithm (it's Eisner (2000) RCDALL algorithm) to see what ranking *would* account for all of the data, or if no ranking of the current constraints would suffice.
7. Automatic analysis simplification by detection of useless constraints (very slow, though).

PHENOMENA COVERED The analysis being to large to cover it in its entirety[4], I will go into the constraints affecting the shape and placement of the reduplicant in some detail, and simply list the other phenomena covered by the analysis.

[4]Complete details in Albro to appear.

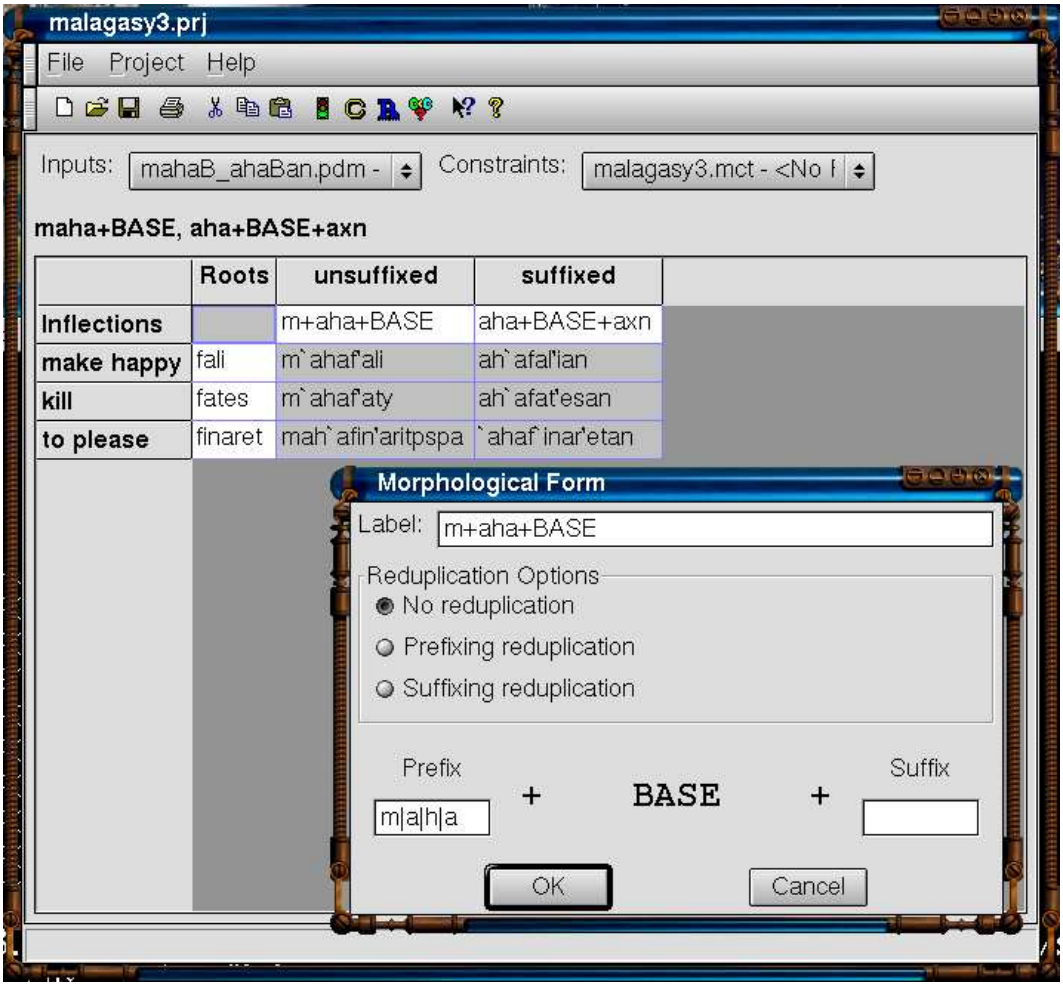

Figure 11: Paradigms

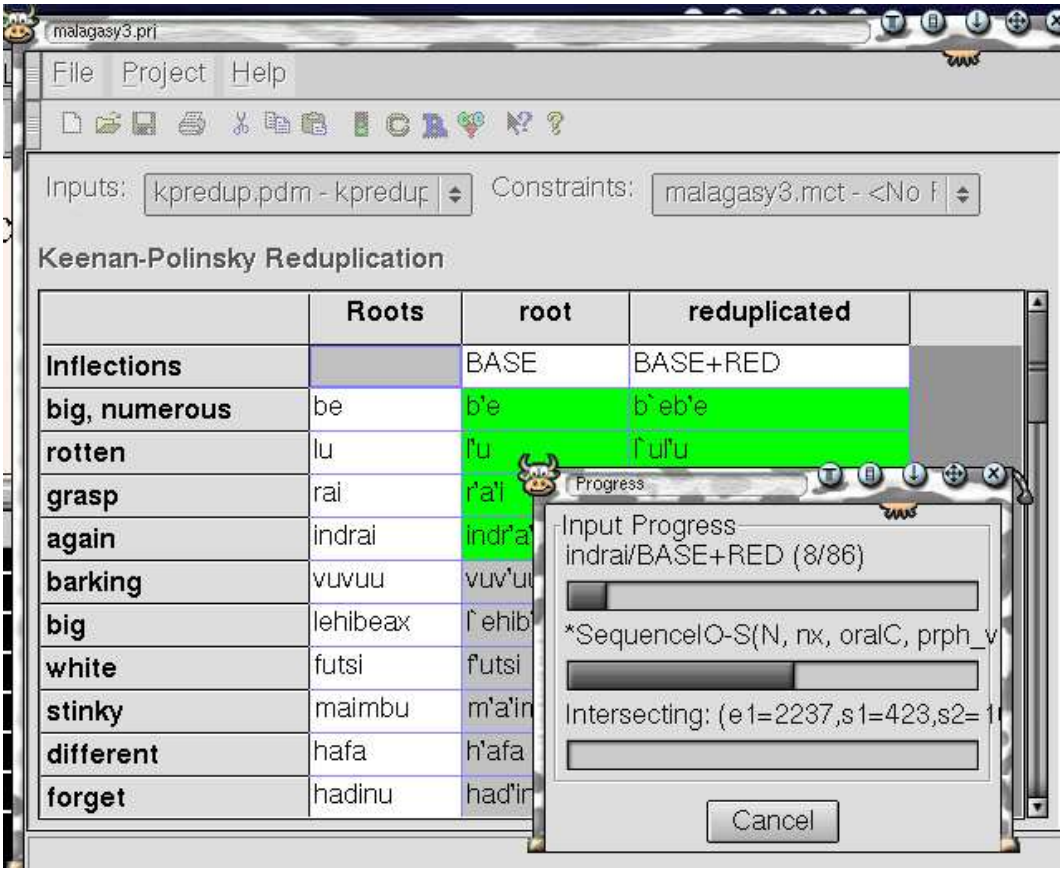

Figure 12: Generation

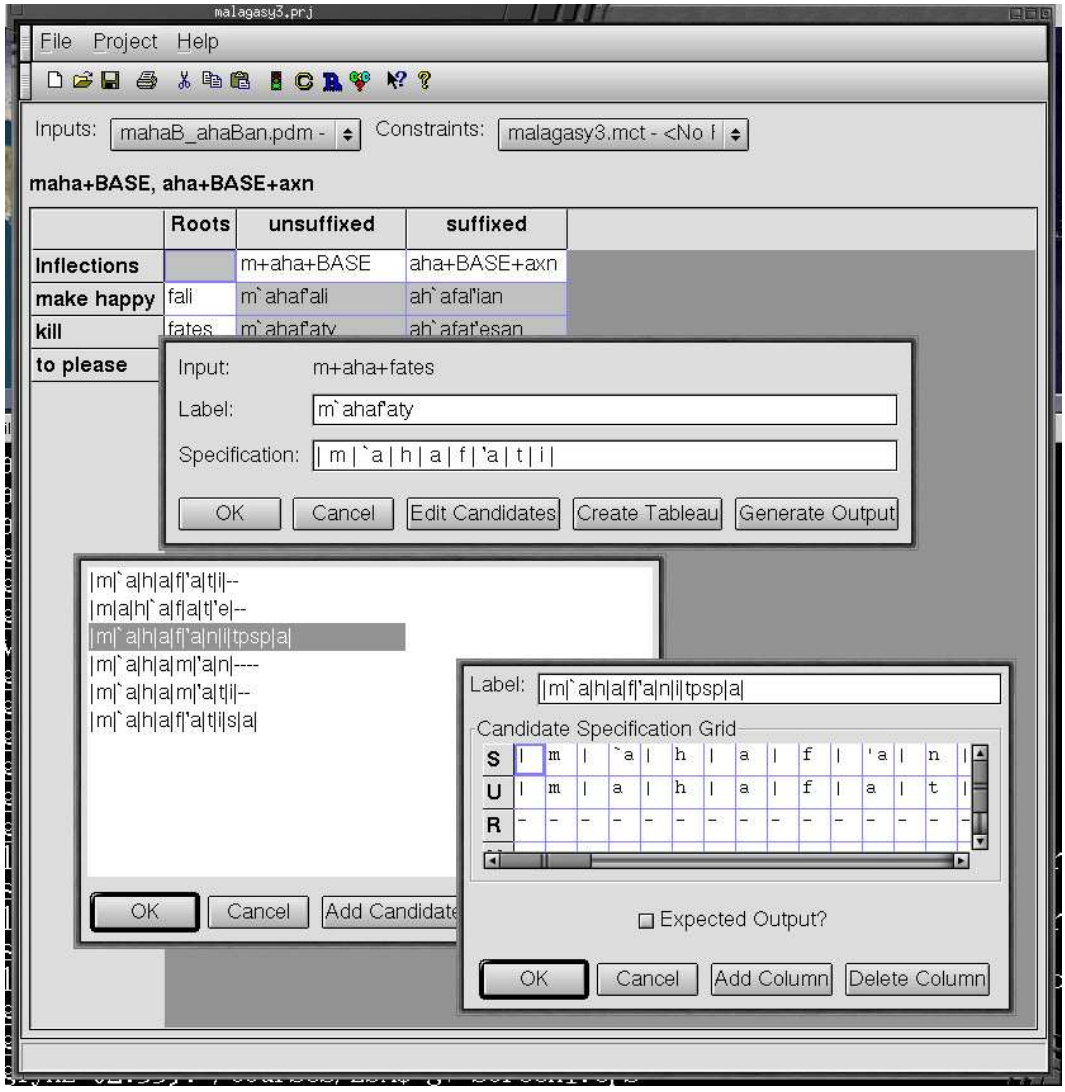

Figure 13: Candidates

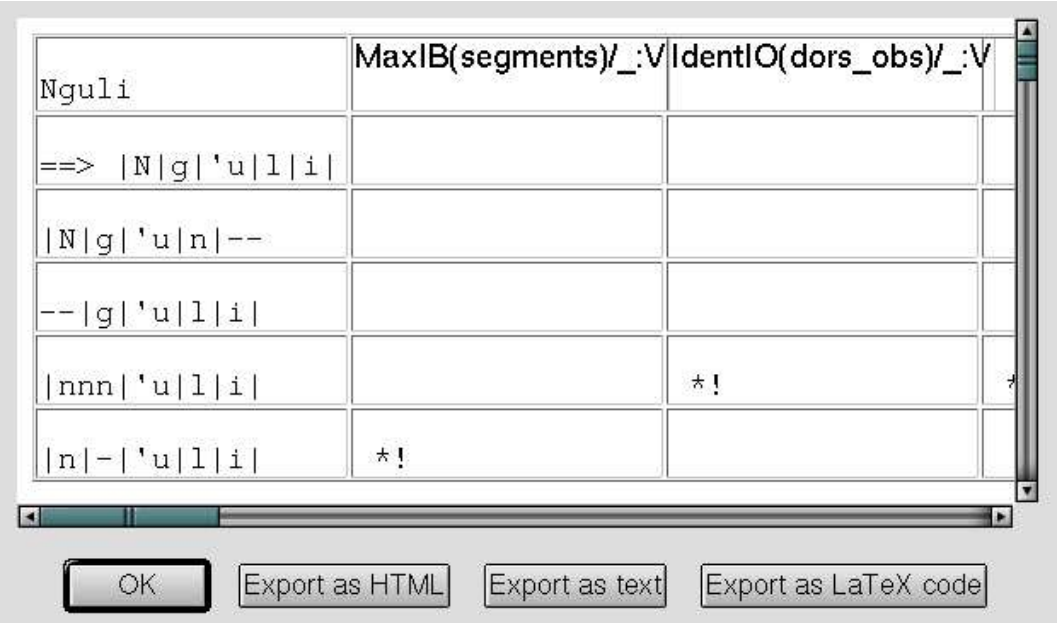

Figure 14: Tableaux

/alahelu+RED/		ExtMaxIR(C)	*(Vps)	PreserveRtM(red)	*InitialM(Vnps, C, red)	*Final(Vs, oralC)	*Clash	MaxBR(C)
☞	|ˌa|l|a|h|ˌe|l|uu|-----|h|ˈe|l|u|		*					*
	|a|l|ˌa|h|e|l|ˌu|---------|l|ˈu|		*			*!	*	**
	|ˌa|l|a|h|ˌe|l|uu|ˌa|l|a|h|ˈe|l|u|		*		*!			
	|ˌa|l|a|h|ˌe|ll|u|-----|h|ˈi|----		*	*!*		*		**
	|ˌa|l|a|h|ˌe|l|uu|ˈa|l|a|h|ˈe|l|u|		**!					
	|ˌa|l|a|h|ˈe|ll|u|----------	*!	*					***

Table 2: Tableau for /alahelu+RED/.

Reduplication Malagasy has suffixing reduplication in which the final foot of a form is reduplicated. The stress pattern of the base is preserved in the reduplicant, occasionally resulting in stress clash otherwise found only in prefixation and compounding. At the base-reduplicant boundary the same hiatus-avoidance and consonant mutation phenomena occur as in compounding, described in the next paragraph. Two illustrative forms are shown in tables 2 and 3. In these tableaux we can see that ExtMaxIR(C) plus PreserveRtM(red) ensure that the rightmost part of the base is preserved in the reduplicant. *(Vps) (a ban on primary stress) keeps from copying a larger portion of the base by using more than one primary stress. *InitialM(Vnps, C, red), in conjunction with the general constraints for the stress pattern of the language, ensures that the reduplicant is the final foot of the word. Finally, IdentBR(Vs) ≫ *Clash ensures that stress patterns are identical between the base and the reduplicant.

Compounding effects At a compounding boundary introduced oral consonant clusters are simplified in favor of the second consonant (preserving only the [-cont] feature of the first consonant). Nasal + oral clusters preserve the nasal consonant, but the nasal assimilates in place to the following consonant, with the following consonant becoming a stop or affricate. Nasal + nasal clusters delete the first nasal.

Stress pattern Malagasy has right-to-left trochaic stress. Primary stress is final in the

/lu+RED/	*InitialM(V_{NPS}, C, RED)	IdentBR(V_S)	*Clash
☞ |l|ˌuˌu|l|ˈu|			*
|l|uu|l|ˈu|		*!	
|l|ˈuˈu|l|u|	*!	*	

Table 3: Tableau for /lu+RED/.

word. Stress is phonemic in some words and prefixes. Stress is attracted to heavy syllables (syllables are heavy in Malagasy if they contain a long vowel or a diphthong), sometimes causing a stress clash. Lapse may occur due to phonemic stress in prefixes and also due to avoidance of footing final epenthetic vowels.

Post-nasal final vowel deletion Vowels that would otherwise be unstressed are not pronounced after a nasal, although the stress pattern acts as though the vowel is there.

/e/ raising Underlying /e/ is pronounced as [i] in syllables after the final stress in a morpheme.

an- prefixation The final nasal of the prefix "an-" (also "man-," and "fan-," which are related) coälesces with following voiceless consonants, emerging as a nasal with the place of articulation of the consonant with which it merged (except in the case of a velar consonant, in which case it emerges as an alveolar). In the case of voiced labials and /h/, merger is also possible, albeit optional. Otherwise the nasal behaves as it would in compounding. Note that final nasals in other prefixes behave differently, following the pattern of compounds.

Weak stems Stems with an underlying final consonant either delete that consonant (for voiced fricatives or alveolar fricatives), or insert a following vowel, changing the consonant to [ʧ], [k], or [n], depending on the nature of the consonant. In reduplication, this epenthetic vowel is inserted after the reduplicant, but is not part of it.

Initial nasal clusters Nasal + oral consonant clusters are pronounced as such initially if the following consonant is voiced. If voiceless, the nasal is deleted.

Hiatus avoidance Hiatus between identical vowels is avoided by deletion of one of the vowels. This is accompanied by compensatory lengthening only in the case of suffixation. Other cases of hiatus are dealt with idiosyncratically, depending on whether the hiatus is induced by prefixation, suffixation, or compounding.

Pre-apocope vowel raising In, some, but not all, words where an underlying consonant is deleted word-finally, preceding underlying /a/ is raised to [i].

4 CONCLUSIONS

The techniques shown here and the toolkit based on them[5] make it possible to carry out large-scale analyses of languages with fairly complex phonological phenomena. This is important, because until analyses of isolated subsets of a language's phonology are assembled into an analysis of the entire language it is impossible to know how the constraints embodied in them will actually interact. In my experience developing this Malagasy analysis, in no case did a set of constraints and rankings developed for some isolated phenomenon fit into the overall system without major changes—usually integration necessitated a more or less complete reanalysis of the phenomenon. A complete language analysis is also valuable because it gives us an idea of what a complete OT grammar might look like. For example, the analysis presented mentions 152 constraints, of which 101 are undominated. It seems likely that other complete language grammars will also have this characteristic wherein the part of the grammatical system that is due to constraint interaction and partially satisfied constraints is in fact dwarfed by the body of constraints which are completely generally applicable (not to mention the nigh-infinite set of constraints which are not applicable at all).

The application of chart parsing to weighted finite state models of OT phonology allows for the first time a computational model adequate to model the empirical data associated with reduplication in a fairly standard Correspondence Theoretic way. This technique also holds promise for a model of Optimality Theoretic syntax.

[5] Available at `http://www.humnet.ucla.edu/people/albro/software.html`, in a version for Microsoft Windows operating systems. The software and its code is also available upon request for Linux, the version from which the screenshots were taken. The code should be directly portable to the Macintosh platform as well, but I have not attempted this.

A RANKINGS

A Hasse diagram showing necessary pairwise rankings for all constraints is shown in Figure 15 , and a smaller Hasse diagram showing just the violable constraints is shown in Figure . . The constraint strata as output by OTPad are as follows:

```
Stratum 0:
   *SplitIO(V)
   *SplitIO(C)
   DepIO(C)
   IdentIOPM(C)
   MaxIB(segments)/_:V
   MaxIR(C)/_V:V
   DepIOBefore(V, V)
   DepIOBefore(V, C)
   DepIOAfter(V, V)
   DepIO(V)/_:V
   DepIO(V)/:V_
   DepIO(V)/_:C
   IdentIO(N)
   DepIOAfter(V, N)
   DepIO(V)/_:N
   SameStress2
   SameStress4
   *CoalescenceIO(V)
   *Contour(C, V, empty)
   *Contour(V, C, empty)
   *oralC C
   *|sonC obs|
   SameStress1
   *|dorsC corC|
   *|corC labC|
   IdentIO(dors_obs)/_:V
   *Contour(Vns_low, Vns_nonlow, empty)
   IdentIO(oral)/_:son-contC
   *|vcl vce|
   *|vcl vce|
   *|-latC lat|
   IdentIOPM(round)
   IdentIO(vce)/_:V
   *(nx)
   *(ax)
   IdentIO([-lat])
   *N N
   *|kg h|
   IdentIO({tp,sp})/:N_
   IdentIO(ant)/_:V
   IdentIO(cor)/_:V
   *|oral nas|
   IdentIO(nonr)/_:V
   *Sequence(e, a, empty)
   *Sequence(e, i, empty)
   *Sequence(a, a, empty)
   MaxIOContour(low, nonlow)
   *oralC#
   IdentIO(lab)/_:V
   *dorsC labC
   *corC labC
   *corC dorsC
   *labC corC
   *dorsC corC
   *labC dorsC
   *antC postC
   *postC antC
   *N contC
   *tp V
   *V sp
   *Final(tp, empty)
   *Initial(sp, empty)
   *|cont stop|
   *|dorsC labC|
   *|corC dorsC|
   *|postC antC|
   *Contour(ant_cons, post_cons, empty)
   *|pb fv|
   *|labC corC|
   *|labC dorsC|
   *Contour(e, i, empty)
   *Contour(i, e, empty)
   *Contour(high, nonhigh, empty)
   Exists(Vs)
   *#N vcl
   MaxIO(nx)/_:C
   ExtMaxIR(C)
   IdentIO([-high])/_:V
   *Final(Vms, not_stressed_V)
   IdentIO(non_e)
   IdentIR(low)
   *SequenceIO-S(N, nx, vcl, oralC, empty)
   *SequenceIO-S(N, nx, oralC,
                 prph_vce_fric, empty)
   IdentIOIn(low, ax)
   IdentIO(front)
   *InitialM(Vnps, C, red)
   DepIR(V)
   *FinalM(unstressed_e,
           not_stressed_V, mbas)
   *FinalM(unstressed_e,
           not_stressed_V, red)
   CompensLgth
   ContiguityM(segments, segments, red)
   IdentIR(vcl)
   IdentIR(dors)
   CompensLgth-N-IB
   CompensLgth-N-IR
   IdentIR(cor)
   *CoalescenceBR(V)
   IdentBR(Vs)
   *SplitBR(segments)
   IdentBR(low)
   IdentBR(V)
Stratum 1:
   *(Vps)
   *SequenceIO-S(N, nx, oralC, vcl, empty)
   *FinalM(periphN, V, zero)
   MaxIB(ax)
Stratum 2:
   IdentIO(obs)
   FMaxIB(obs_stop)
   *SequenceIB(periphN, most, most, V, empty)
Stratum 3:
   MaxIO(oralC)/_:C
   *M(lab_oralC, mbas)
   FMaxIB(fric)/_:V
   IdentIB(vce_fric)
Stratum 4:
   IdentIO(vcl)
   IdentIB(alv_fric)
   IdentIOIn(cor, oralC)
   IdentIO(oral)
   *M(alv_fric, mbas)
Stratum 5:
   MaxIO(N)/_:C
   *FinalSequenceIO-S(nonhigh, most,
```

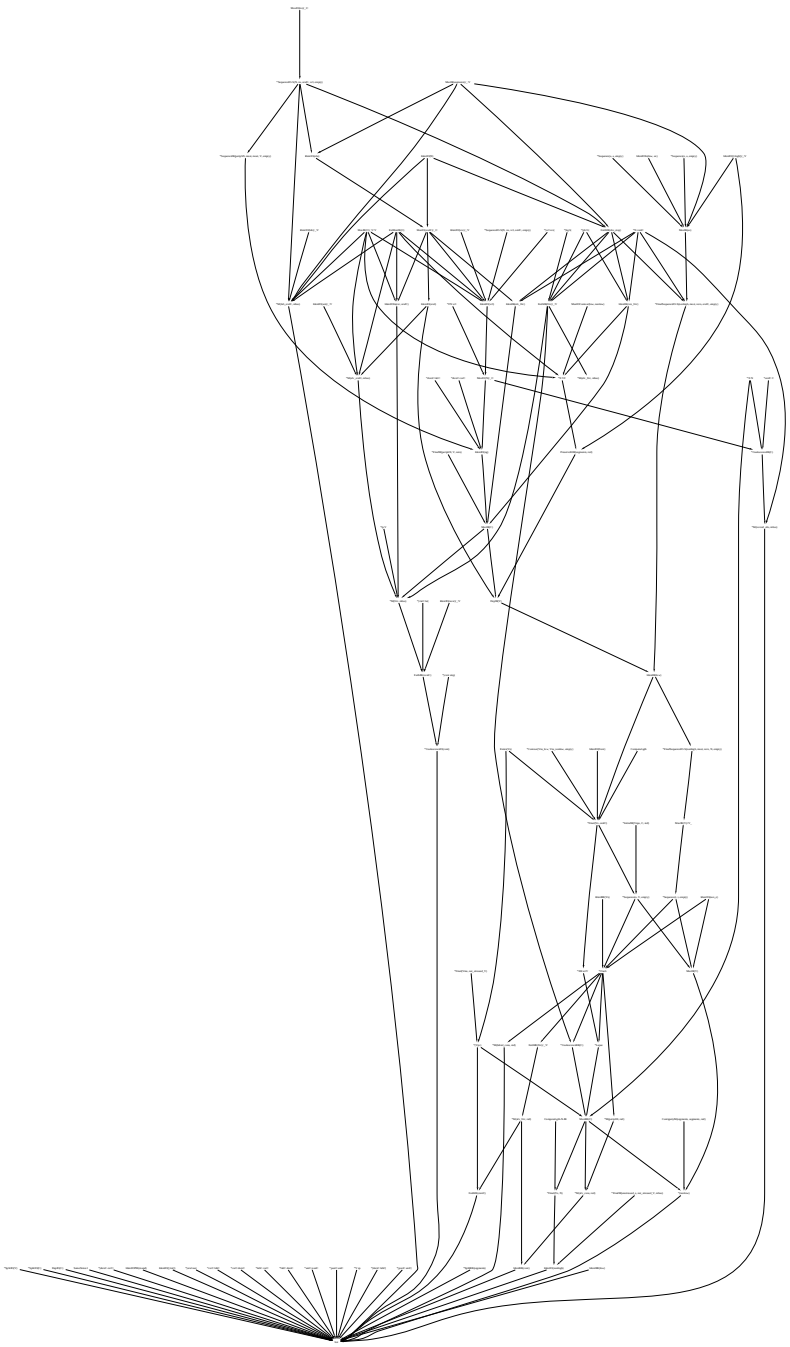

Figure 15: General Hasse Diagram

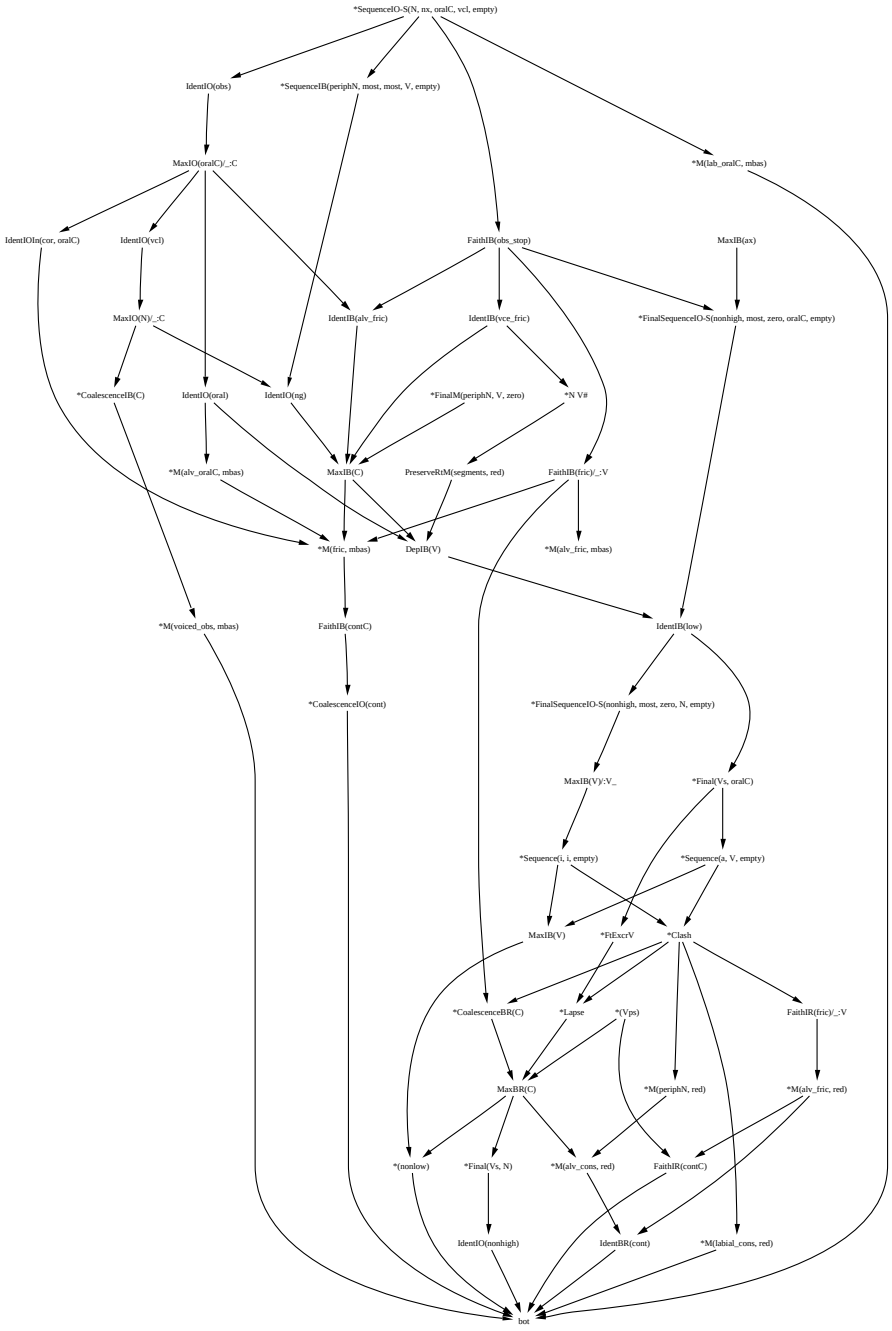

Figure 16: Hasse diagram for violable constraints.

```
                        zero, oralC, empty)
   *M(alv_oralC, mbas)
   *N V#
Stratum 6:
   IdentIO(ng)
   *CoalescenceIB(C)
   PreserveRtM(segments, red)
Stratum 7:
   MaxIB(C)
   *M(voiced_obs, mbas)
Stratum 8:
   *M(fric, mbas)
   DepIB(V)
Stratum 9:
   IdentIB(low)
   FMaxIB(contC)
Stratum 10:
   *FinalSequenceIO-S(nonhigh, most,
                        zero, N, empty)
   *Final(Vs, oralC)
   *CoalescenceIO(cont)
Stratum 11:
   MaxIB(V)/:V_
   *FtExcrV
   *Sequence(a, V, empty)
Stratum 12:
   *Sequence(i, i, empty)
Stratum 13:
   MaxIB(V)
   *Clash
Stratum 14:
   *Lapse
   *CoalescenceBR(C)
   *M(periphN, red)
   *M(labial_cons, red)
Stratum 15:
   MaxBR(C)
   FMaxIR(fric)/_:V
Stratum 16:
   *Final(Vs, N)
   *M(alv_fric, red)
   *M(alv_cons, red)
   *(nonlow)
Stratum 17:
   IdentIO(nonhigh)
   IdentBR(cont)
   FMaxIR(contC)
```

B CONSTRAINT FAMILY DEFINITIONS

The following constraint families are embedded in the OTPad toolkit. Their definitions are given in case they might help to elucidate any mentions above. Note that parameters such as S refer to named natural classes of segments, such as [+NASAL] or C.

*SplitIO(S) A segment matching S 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 S. Versions of this family exist for input-to-reduplicant (IR), input-to-base (IB), and base-to-reduplicant (BR) correspondence as well.

*CoalescenceIO(S) A segment matching 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. Versions of this family exist for input-to-reduplicant (IR), input-to-base (IB), and base-to-reduplicant (BR) correspondence as well.

DepIO(S) For each segment matching 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. Versions of this family exist for input-to-reduplicant (IR), input-to-base (IB), and base-to-reduplicant (BR) correspondence as well. Positional variants exist as well.

For example, DEPIO(S_1)/____S_2 acts like DEPIO(S_1) except that the trigger segment must be followed on the surface by a segment matching S_2 for any violation to accrue.

MAXIO(S) For each segment matching S in the UR, there must exist at least one corresponding segment in the SR. Output a violation for each underlying segment matching S for which there is no corresponding surface segment. Versions of this family exist for input-to-reduplicant (IR), input-to-base (IB), and base-to-reduplicant (BR) correspondence as well. Positional variants exist as well. For example, MAXIO(S_1)/____S_2 acts like MAXIO(S_1) except that the trigger segment must be followed on the underlying form by a segment matching S_2 for any violation to accrue.

FMAXIO(S) (*feature max*) If the a segment matching feature matrix S appears in the underlying form, then a segment matching S must appear on the surface somewhere aligned with it. Outputs a violation for each such matrix appearing in the underlying form without a vertically aligned matching correspondent on the surface. Versions of this family exist for input-to-reduplicant (IR), input-to-base (IB), and base-to-reduplicant (BR) correspondence as well, and also positional variants, which work exactly as in MAXIO.

IDENTIO(S) Given corresponding segments in the UR and the SR, if the underlying segment matches S then the surface segment must match S as well. Output a violation for each mismatched correspondence. Versions of this family exist for input-to-reduplicant (IR), input-to-base (IB), and base-to-reduplicant (BR) correspondence as well. In addition, the family may be specified as IDENTIO($\pm$S) for a bidirectional constraint more similar to the standard IDENTIO family of McCarthy & Prince 1995 (this is like IDENTIO(S) except that additionally if a surface segment matches S then its underlying correspondent, if any, must match S as well). Positional variants exist as well. For example, IDENTIO(S_1)/____S_2 acts like IDENTIO(S_1) except that the trigger segment must be followed on the underlying form by a segment matching S_2 for any violation to accrue.

EXTMAXIO(S) If a segment matching S appears in the UR, a segment matching S must appear in the SR, although they need not correspond. Versions of this family exist for input-to-reduplicant (IR), input-to-base (IB), and base-to-reduplicant (BR) correspondence as well.

*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.

*(S) Ban on S in the SR. Output a violation for each surface segment matching specification S. An extended version of this family exists 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 — $*(\#I^*S)$. An extended version of this family exists where the violation is triggered only for segments within a specified set of morphological domains.

*Final(S, I) Bans S in SR if followed only by members of I — $*(SI^*\#)$. An extended version of this family exists where the violation is triggered only for segments within a specified set of morphological domains.

*Contour(S_1, S_2, I) Within a segment, S_1 may not precede S_2 with only I intervening — $*(S_1 I^* S_2)$.

*Sequence(S_1, S_2, I) A segment matching S_1 may not precede a segment matching S_2 with only segments matching I intervening — $*(S_1 I^* (|I^*)^* S_2)$.

*DoubleSequence(S_1, S_2, S_3, I_1, I_2) S_2 may not appear between S_1 and S_3 with only members of I_1 intervening on the left of S_2 and only members of I_2 intervening on the right of S_2 — $*(S_1 I_1^* | I_1^* S_2 I_2^* | I_2^* S_3)$.

*FinalSequence(S_1, S_2) Like *Sequence, except that it has no intermediate element, and it only notices sequences at the end of words.

Exists(S) Outputs one violation for each word in which no surface segment matches S.

Contiguity(S) Outputs a violation for each instance where two S segments whose underlying correspondents are not contiguous are contiguous on the surface.

PRESERVE{RT,LFT}(S, M) In the case of PRESERVERT, output a violation for each underlying segment matching S, 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. In the case of PRESERVELFT, output a violation for each such underlying segment until the last S segment in the morphological context that has a surface correspondent.

Note that all of these constraint families, given the correspondence relation and representation system described above, can be implemented by weighted finite state machines.

REFERENCES

ALBRO, DANIEL M., 1998. Evaluation, implementation, and extension of Primitive Optimality Theory. Master's thesis, UCLA.

——, to appear. *Phonological Learnability and Analysis in Computational Optimality Theory*. University of California at Los Angeles dissertation.

CHOMSKY, NOAM. 1995. *The Minimalist Program*. Cambridge, MA: MIT Press.

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. 1997. Efficient generation in primitive Optimality Theory. In *Proceedings of the ACL*.

——. 2000. Easy and hard constraint ranking in Optimality Theory: Algorithms and complexity. In *Finite-State Phonology: Proceedings of the 5th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON)*, ed. by Jason Eisner, Lauri Karttunen, & Alain Thériault, 22–33, Luxembourg.

ELLISON, T. MARK. 1994. 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.

Harris, James W. 1969. *Spanish Phonology*. Cambridge, MA: MIT Press.

Hollanger, Frederick S. 1973. *Diksionera/Malagasy-Englisy*. Lutheran Press and the American Cultural Center.

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. 1995. Malagasy morphology: Basic rules. In *The Structure of Malagasy I*, ed. by Matthew Pearson & Ileana Paul, number 17 in UCLA Occasional Papers in Linguistics, 31–48. UCLA Linguistics Department.

——, & ——. 1998. Reduplication in Malagasy. In *The Structure of Malagasy III*, UCLA Working Papers in Syntax and Semantics. UCLA Linguistics Department.

Kurisu, Kazutaka, 2001. *The Phonology of Morpheme Realization*. University of California at Santa Cruz dissertation.

McCarthy, John, & Alan Prince. 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, 2001. Against gradience. Unpublished ms. [Rutgers Optimality Archive #510].

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.

Prince, Alan, & Paul Smolensky. 1993. Optimality Theory: Constraint interaction in generative grammar. Technical Report 2, Center for Cognitive Science, Rutgers University.

Seki, Hiroyuki, Takashi Masumura, Mamoru Fujii, & Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88.

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.