Phonological Reconstruction of a Dead Language Using the Gradual Learning Algorithm

Eric J. M. Smith
Department of Linguistics
University of Toronto
130 St. George Street, Room 6076
Toronto, Ont. M5S 3H1
Canada
eric.smith@utoronto.ca

Abstract

This paper discusses the reconstruction of the Elamite language's phonology from its orthography using the Gradual Learning Algorithm, which was re-purposed to "learn" underlying phonological forms from surface orthography. Practical issues are raised regarding the difficulty of mapping between orthography and phonology, and Optimality Theory's neglected Lexicon Optimization module is highlighted.

1 Introduction

The purpose of this paper is to reconstruct the phonology of Elamite, an extinct language known only from written sources, whose phonology is currently poorly understood. Given that the mechanisms provided by Optimality Theory are powerful enough for a language learner to acquire a natural language given only overt forms, it should be possible to apply the same mechanisms to "learn" Elamite phonology given only its orthography.

The research described here was carried out with the aid of a piece of software, nicknamed *Grotefend*, which was developed as part of a larger research project into Elamite.[1] The data used in this paper consisted of the contents of the *Elamisches Wörterbuch* (Hinz and Koch, 1987) marked up as XML with attributes such as morphology, cognates, semantics, corpus frequency, and chronology. The *Wörterbuch* was used because it is the only source that incorporates Elamite data from all historical periods. It also has the virtue of containing every single attested form known to the authors, which is particularly useful for this project, since we have special interest in alternative spellings of given words.

2 Elamite Language

2.1 Historical and geographical context

Elamite is an extinct language spoken in what is now southwestern and central Iran. Elamite-language texts dating from 2400 BCE until 360 BCE are attested, written in the cuneiform script borrowed from the Sumerians and Akkadians.[2] Elamite has no known linguistic affiliations, although a connection to the Dravidian family has been proposed by McAlpin (1982) and others.

Since both the language and scribal practices are certain to have changed over such a long time-span, this study will restrict itself to text from a single era. The Achæmenid Elamite period (539 BCE to 360 BCE) was chosen, because this period contains the largest volume of texts, and also because those texts are particularly rich in Old Persian names and loanwords that provide a useful starting point for estimating the phonology.

2.2 The cuneiform writing system

As part of their adaptation of cuneiform, the Elamites abandoned most of the logographic ele-

[1] *Grotefend* was written in C++ using Trolltech's Qt toolkit, and runs under Mac OS X. The portions that implement the Gradual Learning Algorithm (§4.3) were adapted from Paul Boersma's Visual Basic source code for the *OTSoft* program, which was kindly provided by Bruce Hayes.

[2] Early texts from Elam using two other indigenous writing systems are not well-enough understood to provide useful linguistic information.

Proceedings of Ninth Meeting of the ACL Special Interest Group in Computational Morphology and Phonology, pages 57–64,
Prague, June 2007.

ments found in Sumerian and Akkadian usage, moving to an almost completely phonetic system, which would be a "core syllabary" in Sproat's (2000) typology. That is, each grapheme represents a syllable, but the system lacks graphemes to represent all of the language's syllables. This is particularly the case for many ⟨CVC⟩ graphemes, which must be written using "syllable telescoping", where a ⟨CV-VC⟩ combination is written, with the internal vowel being repeated. For example, lacking a ⟨lan⟩ grapheme, the syllable /lan/ would have to be written ⟨la-an⟩[3]. Even when the ⟨CVC⟩ grapheme does exist, the ⟨CV-VC⟩ writing is often preferred.

2.3 Hypotheses to be tested

The strategy of this research is to use the techniques of Optimality Theory to reconstruct the language's phonology. We will take the various hypotheses presented by earlier authors and attempt to encapsulate each in the form of an OT constraint.

Altogether 30 separate hypotheses were evaluated, arranged into 11 major groupings. Within each grouping, the sub-hypotheses (which may or may not be mutually exclusive) refer to a related context or related orthographic phenomenon. This paper will largely restrict its discussion to only two of the groupings: H3 (Geminate consonants) and H4 (Nasal vowels).[4]

H3 Geminate consonants
H3a Geminate orthographies represent underlying geminate phonologies.
H3b Geminate orthographies indicate voicelessness (Reiner, 1969).
H3c Certain geminate spellings indicate a distinction other than voicing, such as retroflex/alveolar (McAlpin, 1982).

H4 Nasal vowels
H4a Alternations in the writing of nasals indicate the presence of nasal vowels (e.g. /hũban/ → ⟨hu-um-ban⟩, ⟨hu-ban⟩).
H4b Alternations in the writing of nasals can be explained by underlying nasal consonants (e.g. /humban/ → ⟨hu-um-ban⟩, ⟨hu-ban⟩).

3 Theory of Writing Systems

The discussion of Elamite orthography will be framed within the theory of writing systems proposed by Sproat (2000), whose core claim that "particular (sets of) linguistic elements *license* the occurrence of (sets of) orthographic elements". The details of which linguistic elements license which orthographic ones are specific to any given combination of spoken language and writing system.

The licensing is implemented by a mapping function, $M_{ORL\rightarrow\Gamma}$, whose input is the Orthographically Relevant Level (ORL), and whose output is the orthography(Γ). In the case of Elamite, the relevant level is the surface phonology after the application of phonological processes such as assimilation and cluster simplification, so in Sproat's schema, Elamite is classified as having a "shallow" ORL.[5]

4 Applying OT to Orthography

In the normal application of Optimality Theory, the input and the output are both the same type of linguistic entity. However, in the problem dealt with here, the relationship is between an input that is phonological and an output that is orthographic. The comparison of phonological apples to orthographic oranges leads to complications that will be discussed in §6.1. All the modules of Optimality Theory must be adapted for use with orthography.

4.1 Background

As originally formulated, Optimality Theory can be considered as a set of three interconnected modules: GEN, H-EVAL, and Lexicon Optimization (Prince and Smolensky, 1993). Together GEN and H-EVAL comprise the grammar proper. For any given input, GEN generates a set of output candidates, and these candidates are then evaluated against a set of constraints by the H-EVAL module. Lexicon Optimization is not part of the grammar, but it provides

[3] Or ⟨ 𒆷 𒀭 ⟩, but since the readership of this paper is unlikely to be familiar with cuneiform, all graphemes will be presented in the traditional transliterated form used in Assyriology.

[4] The full list of hypothesis groups also includes: H1 (Interpretation of broken ⟨CV_1-V_2C⟩ writings), H2 (Voicing of stops), H5 (Word-final vowels), H6 (Sibilants), H7 (Existence of an /h/ phoneme), H8 (Existence of an /f/ or /v/ phoneme), H9 (Existence of a /j/ phoneme), H10 (Existence of a /w/ phoneme), and H11 (Existence of an /e/ phoneme). Full discussion of the results for these hypotheses can be found in Smith (2004).

[5] For instance, the noun *kittin* 'length' is spelled ⟨ki-it-ti-imma⟩ when followed by the locative suffix *-ma*, while the 3SG object prefix *in-* is written ⟨id⟩ before a verb like *dunih* 'I gave'.

a mechanism by which language learners can use that grammar to determine underlying forms based on the overt forms that are presented to them.

Optimality Theory can be seen as a model for how a language learner acquires a natural language, presented only with overt forms (Tesar and Smolensky, 2000). At first, the learner's constraint rankings and underlying forms will be inaccurate, but as more information is presented the estimates of the underlying forms become more accurate, which in turn improves the constraint rankings, which further improves the estimates of the underlying forms, and so on. In this study, the "learner" is the *Grotefend* software, which is presented with surface orthography and attempts to deduce the phonology.

4.2 Adaptation for orthography

The GEN module must be adapted to generate plausible overt forms (i.e. rival orthographies). The general strategy for GEN is described in §5, with the specific details given in §5.2.

The constraints are used by a ranking algorithm (§4.3) that compares the rival orthographies from GEN against underlying phonological forms in order to determine the number of constraint violations. In order to start the process, those underlying forms have to be seeded with reasonable initial estimates. If the word is a loanword, the initial estimate is based on the Old Persian or Akkadian phonology. If there is no available loanword phonology, the initial estimate is a direct transcription of the grapheme values as if the word were being read in Akkadian.

Once the constraints have been ranked, Lexical Optimization takes the orthographic forms and the newly-ranked constraints, and calculates an estimated phonology for each of the forms. At this point, the process can stop, or else it can proceed through another iteration of the ranking algorithm, using the new improved estimated phonologies as underlying forms.

4.3 Gradual Learning Algorithm

The Gradual Learning Algorithm (Boersma, 1997; Boersma and Hayes, 2001) is an evolution of the Constraint Demotion algorithm (Tesar, 1995), but avoids the infinite-looping which can arise in Constraint Demotion if underlying forms have more than one overt form. This limitation of Constraint Demotion is a serious one given the data from Elamite orthography; not only are the orthographic forms subject to considerable variation, but also this variation is a key piece of information in attempting to reconstruct the phonology.

In the GLA, constraints each have a numeric ranking value associated with them. It is no longer the case that Constraint A consistently outranks Constraint B; whenever a constraint is evaluated, a random "noise" factor is added to each of the ranking values, and an instantaneous constraint ordering is determined based on these adjusted values. If the ranking values for two constraints are far apart, the noise is unlikely to alter the ordering, and the results will be effectively the same as ordinary OT. If the ranking values for two constraints are close together, the noise could put either constraint on top, but ties are avoided.

In the GLA implementation within *Grotefend*, all constraints start with ranking values of 100.00. With each iteration of the algorithm, one of the observed forms is selected as an exemplar, and rivals (produced by GEN) are compared against the observed exemplar form. Whenever a rival beats the exemplar form, the constraint ranking values must be adjusted: all constraints that picked the wrong winner are penalized (adjusted downwards), and all constraints that picked the right winner are rewarded (adjusted upwards). The size of this adjustment is determined by a variable called "plasticity", which starts at 2.00 and is reduced gradually to 0.002 as the algorithm proceeds through its iterations.

5 Implementation of GEN

The purpose of GEN is to generate a set of plausible overt forms consisting of the real form and a set of rivals which will lose out to the real form. In this problem the overt forms are orthographies, so for any underlying form the challenge is to generate orthographic strings that compete with the real orthography, but which are "wrong" with respect to one or more of the constraints.

5.1 Background

There have been a number of computational implementations of GEN, but the most promising one for our purposes was that of Heiberg (1999). Heiberg's algorithm proceeds by choosing a starting point and then adding constraints to the system. As each constraint is added, new candidates are generated using what she calls "relevant" GEN operations. A GEN operation is considered to be relevant for the current constraint if the operation could affect a candidate's harmony relative to that constraint. So for instance, if the constraint being added evaluates the [+back] feature, the only GEN operations that are relevant are those which affect [+back] or its associations.

The candidates at each stage of the algorithm are not fully formed, and are slowly refined as the constraints are added to the system. One advantage of Heiberg's algorithm is that it functions even if the relative rankings of the constraints are not known. If the constraint rankings are known, the algorithm can operate more efficiently, by culling known losers, but knowing the rankings is not essential.

5.2 Adaptation of GEN for orthography

Generating all "plausible" orthographic candidates for a given form is computationally prohibitive.[6] So, borrowing Heiberg's notion of "relevant" operations, our approach is to generate candidates that specifically exercise one of the constraints in the constraint system.

Each of the hypothesis groups described in §2.3 refers to a particular orthographic context, and each context has a miniature version of GEN to generate appropriately test-worthy rivals. For example, the mini-GEN function for context H3 is as follows:

GEN **H3** (Geminate consonants)
Rule Whenever a geminate consonant is found in the orthography, generate a rival with the non-geminate equivalent.
Example ⟨hu-ut-ta⟩ → { ⟨hu-ta⟩ }

6 Implementation of H-EVAL

The H-EVAL module is responsible for the actual evaluations of the various candidates. It takes any output candidate produced by GEN, and counts the violation marks for each of the constraints.

[6] Initial experiments indicated that a moderately long string of four graphemes would generate in the neighbourhood of 18000 rivals. It seemed unrealistic to evaluate tens of thousands of rivals for each of the 8000+ forms in the database.

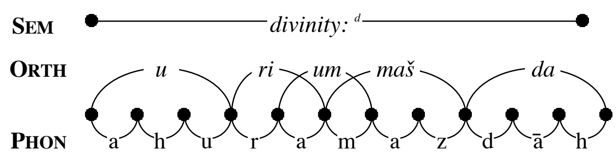

Figure 1: Annotation graph for Ahuramazdāh → ⟨ᵈu-ri-um-maš-da⟩

Each constraint is implemented as a function which takes two inputs (an underlying form and a surface form), and produces as an output the number of violations incurred by the comparing the two inputs. For full generality, both inputs are annotation graphs (Bird and Liberman, 1999; Sproat, 2000) such as the one shown in Figure 1. As implemented in *Grotefend*, the comparison involves only the PHON tier of the underlying form's graph and the ORTH tier of the surface form's graph.

Constraints were written to test each of the hypotheses described in §2.3. Since there is no prior art in the area of constraints involving orthography and phonology, they were developed in the most straightforward way possible. The implementation of these functions is described in §6.2.

6.1 Implementation of alignment

In order to count violations, the two inputs must be properly aligned. For an annotation graph to be "aligned", every grapheme must be licensed by some part of the phonology, and every phoneme must be represented in the orthography. Without such a licensing relationship, it is impossible to make the comparisons needed to count constraint violations.

There is considerable previous work in the area of alignment, most recently summarized by Kondrak and Sherif (2006). The algorithm used in this study is a similarity-based approach, not unlike ALINE (Kondrak, 2000). It differs in some significant respects, notably the use of binary features.

Determining the eligibility of two phonemes for matching requires a distance function. The approach taken was to assign a weight to each phonological feature, and to calculate the distance as the sum of the weights of all features that differ between the two phonemes. The full listing of feature weights is shown in Table 1. The weighting values were deter-

Table 1: Feature weights for computing distance

Phonological Feature	Weight
delayed release, voice, labio-dental, anterior, distributed, strident	1
approximant, continuant, nasal, lateral, round, low, pharyngeal	2
syllabic, consonantal, constricted glottis, spread glottis, high, front, back	4
sonorant, place of articulation	8

mined empirically, selecting the weightings that did the best job of aligning the orthography for the Old Persian loanwords given by Hinz and Koch (1987).

For the actual alignment, several approaches were tried, but the most effective one was simply to line up the consonants and let the vowels fall where they may. For instance, the licensing of ⟨^{d}u-ri-um-maš-da⟩ in Figure 1 used the Old Persian phonology as the best available initial estimate for the Elamite phonology, and proceeded as follows:

⟨**d**⟩ is the divine determinative, and is licensed by the semantic tier of the annotation graph, so it does not need to be anchored to the phonology.
⟨**u**⟩ is anchored at the left edge of the phonology.
⟨**ri**⟩ starts at /r/, but has no clear right edge. The anchoring of /r/ sets a right boundary on the ⟨u⟩, which must therefore be licensed by the initial /ahu/ of /ahuramazdāh/.
⟨**um**⟩ right edge at phoneme /m/; since the ⟨um⟩ has no clear left edge, the second /a/ of /ahuramazdāh/ is left floating between the ⟨ri⟩ and the ⟨um⟩. Since there is no clear choice between the two locations, the /a/ will be shared by ⟨ri⟩ and ⟨um⟩.
⟨**maš**⟩ starts at /m/ and ends at /z/, which is sufficiently similar to match *š*. The /m/ will be shared by ⟨um⟩ and ⟨maš⟩.
⟨**da**⟩ starts at /d/; since ⟨da⟩ is the last grapheme, it must be licensed by the remainder of the phonology.

The general strategy of aligning consonants proved to be an effective one. In the working dataset of Achæmenid Elamite words, there were 3045 that used Old Persian or Akkadian data to provide an initial estimate of the underlying phonology. The algorithm successfully aligned 2902 of those words, for a success rate of over 95%.

6.2 Implementation of constraints

Once the orthography has been successfully aligned with the underlying phonology, it is possible to evaluate the forms for violations against all the constraints in the system. In terms of the tiers shown in annotation graphs like Figure 1, the constraints are performing comparisons between the underlying forms in the PHON tier and overt forms in the ORTH tier. For example, the rules for calculating constraint violations for H3 are as follows:

H3a Geminate spellings indicate geminate pronunciations.
Rule Count a violation if the orthography contains a geminate consonant not matched by a geminate in the phonology.[7]
Violation /a**t**a/ → ⟨a**t-t**a⟩
Non-violations /a**tt**a/ → ⟨a**t-t**a⟩, /a**tt**a/ → ⟨a-**t**a⟩

H3b Geminate spellings indicate voicelessness.
Rule Count a violation if 1) the orthography contains an intervocalic geminate stop not matched by a voiceless stop in the phonology, or 2) the orthography contains an intervocalic non-geminate stop not matched by a voiced stop in the phonology, or 3) the phonology contains an intervocalic voiceless stop not matched by a geminate in the orthography, or 4) the phonology contains an intervocalic voiced stop not matched by a non-geminate in the orthography.[8]
Violations /du**b**aːla/ → ⟨du-i**b-b**a-la⟩, /garma**p**ada/ → ⟨^{d}kar-ma-**b**a-taš⟩
Non-violations /gaumaː**t**a/ → ⟨kam-ma-a**d-d**a⟩, /ba**b**ili/ → ⟨ba-**pi**-li⟩

H3c Certain geminate spellings indicate a distinction other than voicing, such as retroflex/alveolar.
Rule Count a violation if 1) the orthography contains a ⟨Vl-lV⟩ or ⟨Vr-rV⟩ sequence not matched by a retroflex in the phonology, or 2) the phonology contains a /ɭ/ or /ɽ/ not matched by a ⟨Vl-lV⟩ or

[7]The claim made by Grillot-Susini and Roche (1988) was only that a geminate orthography represents a geminate phonology; a non-geminate orthography could still conceal a geminate phonology.

[8]Reiner (1969) restricted her claim about gemination representing voicelessness to intervocalic stops. Word-initial stops and intervocalic non-stops were not relevant here.

⟨Vr-rV⟩ in the orthography.
Violations /ta**l**u/ → ⟨ta-a**l**-**l**u⟩, /ta**ɭ**u/ → ⟨ta-**l**u⟩
Non-violation /ta**ɭ**u/ → ⟨ta-a**l**-**l**u⟩

7 Implementation of Lexicon Optimization

Given the set of constraints provided in §6.2 and rankings determined by the Gradual Learning Algorithm (§4.3), it is now possible to move on to the final stage of the "learning" process: Lexicon Optimization, which is responsible for choosing the most harmonic input form for any given output form.

There has been surprisingly little literature devoted to Lexicon Optimization, and discussions of how it might be implemented have been restricted to toy algorithms such as Itô et al's (1995) "tableau des tableaux". Hence, a novel approach was devised, based on the observation that Lexicon Optimization is a sort of mirror image of H-EVAL. For H-EVAL, there exists a separate GEN module whose task is to generate the possible output candidates. Clearly, Lexicon Optimization needs an equivalent module, but one that would generate a range of rival input forms. Since the GEN algorithm described in §5.2 uses a constraint-driven technique for generating output candidates, it seems appropriate to also use a constraint-driven technique for generating input candidates. Accordingly, this anti-GEN is implemented as a set of miniature anti-GENs, each of which is responsible for generating "relevant" input candidates for one of the hypothesis groupings. For example, the anti-GEN function for H3 is as follows:

Anti-GEN **H3** (Geminate consonants)
Rule Whenever a geminate consonant is found in the orthography, create input candidates with the geminate phonology and the equivalent non-geminate phonology. If the geminate orthography is a ⟨Vl-lV⟩ or ⟨Vr-rV⟩, also create an input candidate with a "retroflex" phonology.[9]
Example ⟨ta-al-lu⟩ → { /tallu/, /talu/, /taɭu/ }

8 Results and Discussion

We ran 40000 iterations of the Gradual Learning Algorithm against the Achæmenid Elamite forms found in the *Elamisches Wörterbuch*. The final constraint rankings for hypothesis groups H3 and H4 are shown in Table 2.[10] The combination of constraints, GEN, and anti-GEN functions used by *Grotefend* tends to penalize constraints much more often than it rewards them. The absolute ranking values are not significant; what matters is their relative values.

Table 2: Final constraint rankings for H3 and H4

Hypothesis	Constraint	Ranking Value
H4b	NasalConsonants	−136.93
H3b	⟨Geminate⟩=/Voiceless/	−283.70
H4a	NasalVowels	−1434.77
H3c	⟨Geminate⟩=/Retroflex/	−1629.74
H3a	⟨Geminate⟩=/Geminate/	−3189.11

8.1 Results for H3 (Geminate consonants)

The results for H3 strongly support the hypothesis (Reiner, 1969) that geminate orthographies are an attempt to indicate voicelessness; the opposing hypothesis (Grillot-Susini and Roche, 1988) that geminate orthographies represent geminate phonologies ended up being very heavily penalized.

What was surprising was that hypothesis H3c, that ⟨Vl-lV⟩ and ⟨Vr-rV⟩ geminates represent a separate phoneme from the non-geminate orthographies, ranked so poorly. The problem here is a side-effect of the process for generating input candidates.

Consider the Akkadian name *Nabû-kudurri-uṣur*; the ⟨ur-ri⟩ sequence that occurs in the various spellings of this name would appear to be an ideal context for evaluating H3c. However, when generating input candidates for *Nabû-kudurri-uṣur*, the various anti-GEN functions create 238 permutations (mostly permutations of voicing), but only four of those input candidates contain an /ɽ/ phoneme, with the rest having an /rr/ or an /r/. Since the anti-GEN function produces so few /ɽ/ input candidates for the ⟨ur-ri⟩ orthography, it is likely that the software will find an /r/ in the underlying phonology, and will count a violation against this constraint.

The prejudice against /ɭ/ and /ɽ/ highlights the importance of having a fair and balanced anti-GEN

[9] McAlpin (1982) hedges on whether the phonology represented by these geminates actually represents retroflexion, but he then proceeds to discuss Proto-Elamo-Dravidian cognates as if this orthography actually did represent a retroflex articulation.

[10] Results for the other nine groups are in Smith (2004).

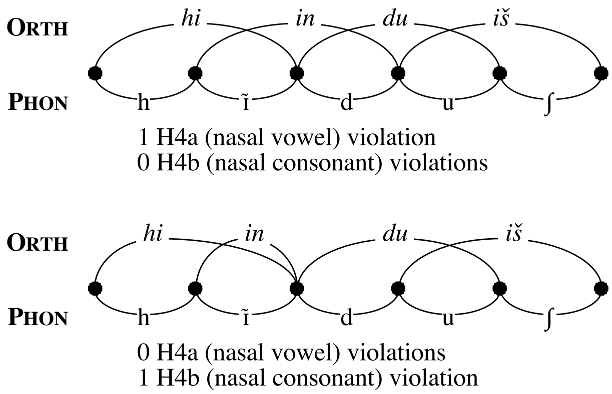

Figure 2: Licensing of ⟨hi-in-du-iš⟩ 'Indian'

function. The proposal to be discussed in §8.3 for cross-permuting the results of the constraint-specific anti-GEN functions would probably also improve the results for this hypothesis.

8.2 Results for H4 (Nasal vowels)

The effectiveness of the constraints for evaluating nasals was undermined by choices made in the alignment algorithm. Although constraint H4b (NasalConsonants) is ranked significantly higher than H4a (NasalVowels), this may be merely a side-effect of the alignment algorithm.

Consider the word for 'Indian', which shows up as ⟨hi-du-iš⟩, ⟨hi-in-du-iš⟩, or ⟨in-du-iš⟩. It is not unreasonable to postulate an underlying phonology of /hĩduʃ/, based both on the range of written forms, and on the Old Persian phonology. However, when the alignment algorithm attempts to determine which phoneme sequences are licensing which graphemes, it has a difficult choice to make for the ⟨in⟩ grapheme. Licensing the vowel portion of ⟨in⟩ is straightforward, but what should be done for the consonant? If the software assumes that the most salient features are [+consonantal] and [−syllabic], we get the first annotation graph shown in Figure 2, but if the most salient features are [+nasal] and [+sonorant], we get the second graph.

The choice of how to license the ⟨in⟩ grapheme makes a difference for how the H4a and H4b constraints are evaluated. Using the weightings given in Table 1, the software will align ⟨in⟩ with /ĩd/, because the distance between /n/ and /d/ is less than that between /n/ and /ĩ/. Hence, the alignment algorithm chooses the first of the two annotation graphs given in Figure 2. This has the result of prejudicing the learning algorithm in favour of H4b instead of H4a. Ideally, the alignment algorithm should be neutral with respect to the various constraints.

The licensing of the ⟨in⟩ sign in this example is one case of several where it appears that using phonological segments as the basis for licensing may be the wrong thing to do. It would be better to think of the second portion of the ⟨in⟩ sign in ⟨hi-in-du-iš⟩ as being licensed by a [+nasal] feature, without attempting to tie the feature down to either the /i/ or the /d/ segment.[11]

8.3 Discussion of Lexicon Optimization

The generation of useful input candidates is limited by the information that is available to us. For all we know, Elamite had an /ɯ/ vowel, and *Grotefend* could even generate input candidates that contained an /ɯ/. However, none of the constraints would weigh either for or against it, so there is no point in generating such an input candidate. Consequently, the correct underlying form may well be inaccessible to Lexicon Optimization. At best, Lexicon Optimization can produce an estimated underlying form that leaves as underspecified any features that cannot be verified by a corresponding constraint. This is a limitation of Lexicon Optimization in general, not just of the implementation in *Grotefend*.

One problem specific to our constraint-based generation of input candidates is that the anti-GEN functions work in isolation from each other. For example, when processing ⟨da-iš⟩, the H1 (broken-vowel) anti-GEN produces /daiʃ/, /dajʃ/, /dɛʃ/, and /daʃ/. Separately, the H6 (sibilant) anti-GEN will produce /dais/, /daiʃ/, /daiz/, /daitʃ/, and /daits/. Since the two functions operate independently, the software fails to generate a whole range of candidates. If the actual underlying phonology were /dɛtʃ/, *Grotefend* would never find it, since that particular phonology will never be generated and presented to Lexicon Optimization as a possible input candidate. A more sophisticated anti-GEN implementation would allow for the input candidates produced by one constraint's anti-GEN function to be further permuted

[11] Sproat (2000) uses phonological segments to describe licensing, but there is nothing in his theory that requires this; in fact, he says that his use of segments is merely a "shorthand" for a set of overlapping gestures.

by the anti-GEN function of another constraint.

9 Conclusions

This project represented an expedition into three largely unexplored territories: the application of Optimality Theory to orthography, the implementation of Lexicon Optimization in software, and the mass analysis of Elamite phonology. All three presented unanticipated challenges.

The problem of implementing GEN algorithmically appears to be at an early stage even in the processing of phonological data. The constraint-driven GEN adopted from Heiberg (1999) does appear to be a useful starting point for working with orthography.

The determination of the mapping between phonology and orthography can have unexpected consequences for the evaluation of constraints. Even when properly aligned, implementing meaningful constraints to evaluate the mismatches between phonology and orthography proved to be surprisingly complex. An alternative representation, licensing graphemes based on bundles of features rather than phonemes, might be more effective.

The whole area of Lexicon Optimization has received surprisingly little mention in the literature of Optimality Theory. The notion that there must be some form of anti-GEN module to produce suitable input candidates appears never to have been raised at all. The existence of anti-GEN is hardly specific to the study of orthography, but would seem to be an omission from Optimality Theory in general.

The constraint-driven implementation of the anti-GEN function does seem like a promising strategy, although the details need work. In particular, there is a need for the outputs of the various constraint-specific anti-GENs to be permuted together in order to produce all plausible input candidates.

Elamite has always been problematic both due to its status as an isolate and because the available clues end up being obscured by the writing system. So far, we can claim that this computational analysis of the body of Elamite vocabulary has succeeded in duplicating some of the tentative conclusions drawn from a century of hard work "by hand".

References

Steven Bird and Mark Liberman. 1999. A formal framework for linguistic annotation. Technical Report Technical Report MS-CIS-99-01, Department of Computer and Information Science, University of Pennsylvania.

Paul Boersma and Bruce Hayes. 2001. Empirical tests of the gradual learning algorithm. *Linguistic Inquiry*, 32:45–86.

Paul Boersma. 1997. How we learn variation, optionality, and probability. *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam*, 21:43–58.

Françoise Grillot-Susini and Claude Roche. 1988. *Eléments de grammaire élamite*. Etudes elamites. Editions Recherche sur les civilisations, Paris.

Andrea Heiberg. 1999. *Features in Optimality Theory: A computational model*. Ph.D. thesis, University of Arizona.

Walther Hinz and Heidemarie Koch. 1987. *Elamisches Wörterbuch*. D. Reimer, Berlin.

Junko Itô, Armin Mester, and Jaye Padgett. 1995. NC: Licensing and underspecification in optimality theory. *Linguistic Inquiry*, 26(4):571–613.

Grzegorz Kondrak and Tarek Sherif. 2006. Evaluation of several phonetic similarity algorithms on the task of cognate identification. In *Proceedings of the Workshop on Linguistic Distances*, pages 43–50.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proceedings of the First Meeting of the NAACL*, pages 288–295.

David W. McAlpin. 1982. Proto-Elamo-Dravidian: The evidence and its implications. *Transactions of the American Philosophical Society*, 71(3):1–155.

Alan Prince and Paul Smolensky. 1993. Optimality theory. *Rutgers Optimality Archive*, #537.

Erica Reiner. 1969. The Elamite language. *Handbuch der Orientalistik I/II/1/2/2*, pages 54–118.

Eric J. M. Smith. 2004. *Optimality Theory and Orthography: Using OT to Reconstruct Elamite Phonology*. M.A. forum paper, University of Toronto.

Richard Sproat. 2000. *A Computational Theory of Writing Systems*. Cambridge University Press.

Bruce Tesar and Paul Smolensky. 2000. *Learnability in Optimality Theory*. MIT Press, Cambridge, Mass.

Bruce Tesar. 1995. *Computational Optimality Theory*. Ph.D. thesis, University of Colorado.