Learning Phonotactic Grammars from Surface Forms

Jeffrey Heinz
University of California, Los Angeles

1. Introduction

This paper presents an unsupervised batch learning algorithm for phonotactic grammars without a priori Optimality-theoretic (OT) constraints (Prince and Smolensky 1993, 2004). The underlying premise is that linguistic patterns (such as phonotactic patterns) have properties which reflect properties of the learner. In particular, this paper explores one way in which a learner benefits from considerations of locality; i.e. "the well-established generalization that linguistic [here: phonological] rules do not count beyond two" (Kenstowicz 1994:597).[1] The formalization of locality adopted here leads to a novel, nontrivial hypothesis: all phonotactic patterns are neighborhood-distinct (to be defined in §5).

1.1 Background

The foundational work in learning OT grammars (Tesar 1995, Tesar and Smolensky 1998) solves the problem of determining the constraint ranking given a set of constraints, underlying forms and candidate sets. Much work builds on this foundation (Boersma 1997, Tesar 1998, Hayes 1999, Boersma and Hayes 2001, Pater and Tessier 2003, Pater 2004, Prince and Tesar 2004, Hayes 2004, Riggle 2004, Alderete et al. 2005, Merchant and Tesar to appear, Wilson 2006, Tessier 2006, Boersma 2006). Recent research also shows that learning is possible even when the learning data is small relative to the possible number of grammars (Lin 2002, Riggle in preparation). Despite these welcome results, there has been growing interest in algorithms that can learn grammars without underlying forms (Alderete et al. 2005, Jarosz 2006, Merchant and Tesar to appear, Riggle 2006) and without *a priori* constraints.

Hayes (2004) motivates a 'pure phonotactic learner' by hypothesizing that a complete phonological grammar is more easily learned in stages: the phonotactics of the target language are learned prior to the alternations. This strategy follows from the observation that children appear to acquire alternations later than the phonotactics, much of which appears to be in place by as early as one year. The learning model in Albright and Hayes (2002, 2003), for example, succeeds in part because it has been given phonotactic constraints that cued rule discovery. (See Gildea and Jurafsky (1996) for another study of rule induction.)

There is a growing body of phonotactic learning algorithms, including Ellison (1991), Frisch (1996), Coleman and Pierrehumbert (1997), Frisch et al. (2004), Albright (2006), Goldsmith (2006) and Hayes and Wilson (2006). Unlike many of these works which propose stochastic phonotactic models, this algorithm returns only categorical generalizations. Gradient phonotactic models are often evaluated by analyzing how the mature, learned grammar matches humans' performance on 'wug' tests pioneered by Berko (1958). This learner, inspired by Angluin (1982), is evaluated by demonstrating that the target grammar is returned by the learner given a sufficient word sample generated by the target grammar.[2]

In §2, I introduce the case studies under consideration. In §3, I define the algorithm and illustrate it with one of the case studies. §4 discusses the results of the algorithm. In §5, I introduce a class of languages and the neighborhood-distinct hypothesis.

2. Three Case Studies

There are three pseudo-languages considered in this paper: a language in which the value of the

*I especially thank Bruce Hayes, Ed Stabler, Colin Wilson, and Kie Zuraw for their insights and suggestions. I also thank Sarah Churng, Greg Kobele, Katya Pertsova, Sarah Van-Wagnenen, and members of the UCLA Phonology Seminar for helpful discussion.

[1] See also McCarthy and Prince (1986).

[2] See Osherson et al. (1986), Kearns and Vazirani (1994) for overviews of formal language learning theory.

 Proceedings of the 25th West Coast Conference on Formal Linguistics, ed. Donald Baumer, David Montero, and Michael Scanlon, 186-194. Somerville, MA: Cascadilla Proceedings Project.

[ATR] feature predictably follows from the first vowel (e.g. Kalenjin (Tucker 1964, Lodge 1995) (see also Baković (2000) and references therein), a language in which the [ATR] feature contrasts everywhere (e.g. Akan (Stewart 1967, Ladefoged and Maddieson 1996, Archangeli and Pulleyblank 1994)), and a language where the value of the [ATR] feature is predictable based on whether it is in a closed (-) or open (+) syllable (e.g. Javanese (Archangeli 1995)). I will refer to these languages as the ATR Harmony Language, the ATR Contrastive Language and the ATR Allophony language, respectively.

To simplify matters, I assume all languages only allow CV(C) syllables, though word-initially vowels are acceptable. I assume each language has ten vowels in which {i,u,e,o,a} are [+ATR] and {I,U,E,O,A} are [-ATR]. Each language has eight consonants {p,b,t,d,k,g,m,n}. Vowels are [+syllabic] and consonants are [-syllabic] and have no value for [ATR].

Thus, each language has two phonotactics. They all share the same syllable structure phonotactic, but the distribution of the feature [ATR] differs for each language. I illustrate the learning algorithm with respect to the ATR Harmony language, and give the results for the other languages in the appendix.

3. The Learner

The learner represents a phonotactic grammar as a finite state machine. Because a finite state machine accepts or rejects words, it meets the minimum requirement for a phonotactic grammar—a device that at least answers Yes or No when asked if some word is possible (Chomsky and Halle 1968, Halle 1978). Also, they can be related to finite state OT models, which allow one to compute a phonotactic finite state acceptor (Riggle 2004), which becomes a target grammar for the learner. Finally, they are well-defined and can be manipulated (Hopcroft et al. 2001).[3]

In the broadest terms, the learner operates by writing smaller descriptions of the observed forms guided by the notion of natural class and a structural notion of locality. Specifically, it operates in three steps. First, it builds a structured representation of the input using natural classes. Second, it generalizes using an abstract notion of locality I call *the neighborhood*. Third, it checks for distributional dependencies to factor out redundancy in the representation. These steps are repeated with natural classes that pick out fewer and fewer segments, the most general natural classes being considered first.

3.1 Structuring the Input

Consider the ATR Harmony Language, where vowels in each word agree in [ATR]. The table in (1) provides a sample of words from this language.

(1)

1.	a	6.	atapi	11.	ikop	16.	kUtEpA	21.	kOn
2.	ka	7.	bedko	12.	eko	17.	pOtO	22.	pAtkI
3.	puki	8.	piptapu	13.	I	18.	AtEtA	23.	kUptEpA
4.	kitepo	9.	mitku	14.	kO	19.	IkUp	24.	pOtkO
5.	pati	10.	etiptup	15.	pAkI	20.	Ak	25.	AtEptAp
								26.	IkU

How can a finite state acceptor be acquired from a finite list of words like the ones above?

The input is structured by partitioning the segmental inventory by natural class and constructing a prefix tree. A partition by natural class is a collection of sets of segments which are disjoint but whose union is the segmental inventory and each which forms a natural class. For example, π_0 below uses the feature [syllabic] to divide the inventory into two non-overlapping groups. π_1 divides the inventory into three non-overlapping groups using the features [+syllabic,+ATR], [+syllabic,-ATR] and [-syllabic].

(2) $\pi_0 = \{i, u, e, o, a, I, U, E, O, A\} \mid \{p, b, t, d, k, g, m, n\}$

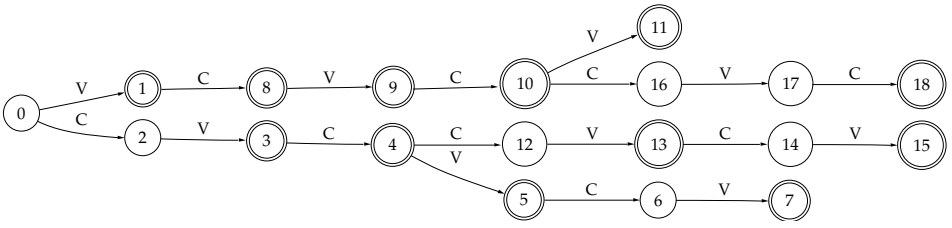
(3) $\pi_1 = \{i, u, e, o, a\} \mid \{I, U, E, O, A\} \mid \{p, b, t, d, k, g, m, n\}$

The partitions by natural class act as a lens the learner wears when observing a surface form. For example, [mitku] is read as [CVCCV] by π_0.

[3] See also Johnson (1972), Kaplan and Kay (1994), Ellison (1991), Eisner (1997b,a, 2000), Albro (1998, 2005), Karttunen (1998, 2006) and Riggle (2004) for finite-state approaches to phonology.

Constructing a prefix tree is a standard algorithm (Angluin 1982). A prefix tree is built one word at a time. As each word is added, an existing path in the machine is pursued as far as possible. When no further path exists, a new one is formed. The prefix tree for the words in (1) using π_0 is in (4).[4]

(4)



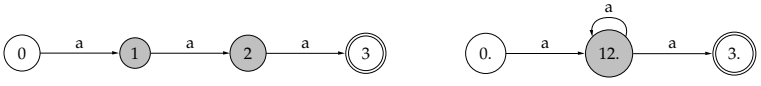
This is a structured representation of the input forms. The learner has already made some generalizations by structuring the input with the [syllabic] partition π_0—e.g. the current grammar can accept any CVCV word. However, the current grammar undergeneralizes because it cannot accept words of four syllables like CVCVCVCV. It also overgeneralizes because it can accept a word like *bitE*.

3.2 State Merging

This section aims to correct the undergeneralization by merging equivalent states in the prefix tree. State-merging is a process where two states are identified as equivalent and then *merged* (i.e. combined). A key concept behind state merging is that transitions are preserved (Hopcroft et al. 2001, Angluin 1982). This guarantees that the post-merged machine always accepts every word the pre-merged machine accepts, though the post-merged machine may accept more.

To illustrate, consider the machine on the left hand side of (5). When states 1 and 2 are merged, transitions must be preserved. Consequently, in the merged machine, shown on the right, there is a transition from state 12 to itself—this is the transition from state 1 to state 2 in the pre-merged machine. The machine on the left only accepts one word $\{aaa\}$, but the post-merged machine accepts an infinite number of words $\{aa, aaa, aaaa, \ldots\}$.

(5)



Which states are chosen to be merged determines the kinds of generalization that can be made. This learner merges states iff their immediate environment is the same. This environment is the *neighborhood*. It is:

(6) a. the set of incoming symbols to the state
b. the set of outgoing symbols to the state
c. whether it is final or not.

As an example in (7), states p and q have the same neighborhood because they are both nonfinal states, they can only be reached by the same set of symbols {a,b}, and they can only be departed from by the same set of symbols {c,d}.

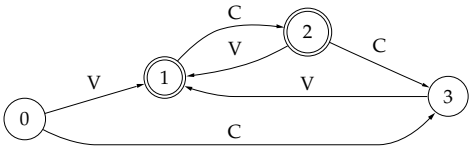
(7)

The learner merges states in the prefix tree with the same neighborhoods. For example, states 4 and 10 in (4) have the same neighborhood. So these states are merged. When all states with the same

[4]For all machines in this paper, I mark the start state with 0, and final states with double peripheries.

neighborhood have been merged in the prefix tree the result is given in (8).[5]

(8)



The machine in (8) accepts the words V, CV, CVC, VCV, CVCV, CVCVC, CVCCVC, ... In fact, the learner has generalized exactly in the manner desired; i.e. it has acquired the syllable structure phonotactic. Note there is still overgeneralization because the ATR vowel harmony constraint has not been learned (e.g. *bitE* can be accepted by the machine in (8)).

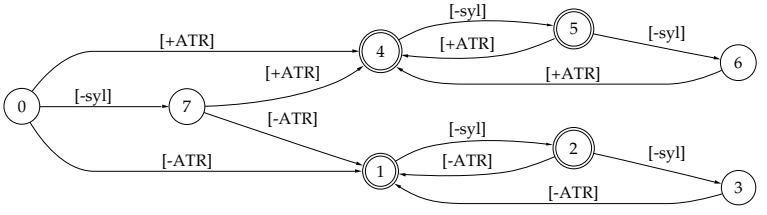
So far, learning has proceeded in two steps. A prefix tree is constructed using some partition by natural class. States with the same neighborhood in the prefix tress are then merged.

3.3 Distributional Dependencies

This section explains how the learner decideds to exclude words like *bitE* from being generated by the grammar. This step involves comparing the machine just acquired to ones previously acquired and factoring out redundancy. Since the learner has acquired only one machine at this point, this step is skipped in the first loop of the algorithm.

In subsequent loops, the algorithm repeats the prefix tree construction and state-merging steps using a finer partition by natural class. I illustrate by using partition π_1 above. When the prefix tree is built with π_1 and the states with the same neighborhood are merged, the result is shown in (9).

(9)



Every word this machine accepts does not contain vowels with different values for [ATR]. The learner has the right language—again, just by structuring the input and merging states with the same neighborhood, the learner makes the right generalization. However, the machine redundantly encodes the syllable structure. Now it is possible to introduce the third step of the learner, which aims to factor out redundancy by checking for distributional dependencies.

In particular, the learner needs to know if the distribution of the [ATR] features depends on the distribution of consonants. The intuition is to ask if the vocalic paths in the syllable structure machine in (8) are traversed by both [+ATR] and [-ATR] vowels. This is accomplished by carrying out the following. First remove [+ATR] vowel transitions from the machine in (9), replace the [-ATR] labels with [+syllabic] labels, and check whether the resulting acceptor accepts the same language as the syllable structure acceptor.[6] Do the same with [-ATR] vowels. If it does in both instances then the distribution of ATR is independent of [-syllabic]. Otherwise, the distribution of [ATR] depends on [-syllabic].

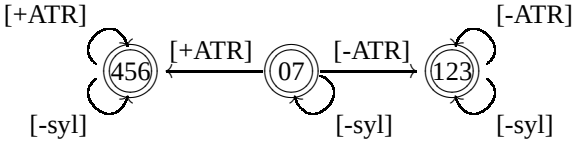
If the distribution of [ATR] is independent of [-syllabic], the states connected by the [-syllabic] transition are merged. The idea here is that observing a [-syllabic] element is not something that needs to be recorded when considering the distribution of [ATR]. If the distribution of [ATR] depends on [-syllabic], make two machines, one by merging states connected by transitions with the [+ATR] label, and one by merging those with the [-ATR] label.

[5]The machines shown after merging have been minimized for readability—a minimized machine is the smallest deterministic machine which accepts the same language (Hopcroft et al. 2001).

[6]This is possible by checking if the machines' minimized versions are identical (Hopcroft et al. 2001).

In this case, the check above tells the learner that [ATR] is independent of [-syllabic]. Thus, the learner merges states in (9) that are connected by [-syllabic] transitions. Thus states 0 and 7 are merged, states 4, 5, and 6 are merged and states 1, 2, and 3 merged. The result is given in (10).

(10)



The learner has acquired the vowel harmony constraint.

3.4 Summary of the Learner

The steps followed by the learner are repeated in (11).

(11)
1. Build a prefix tree of the sample with some partition.
2. Merge states with the same neighborhood.
3. Compare this machine to one acquired earlier under some coarser partition by natural class.
   (a) If the refined blocks in the partition are independent of the static blocks, merge states that are adjoined by static blocks.
   (b) If not, make two machines by merging states adjoined by the refined blocks.
4. Repeat 1-3 with the next partition until all partitions by natural class are exhausted.

The partitions by natural class are ordered so that the learner considers the coarser partitions before finer ones. The algorithm incrementally returns individual finite state machines each which encodes some regularity about the language. Each individual machine is a phonotactic pattern and a surface-true constraint. A phonotactic grammar is the set of these machines, all of which must be satisfied simultaneously for a word to be acceptable (i.e. the intersection of all the machines is the actual grammar). Thus, although the vowel harmony phonotactic shown in (10) accepts words with hundreds of adjacent consonants, this is not a permissible word in the ATR Harmony language because the syllable structure phonotactic prohibits it.

4. Results and Discussion

The algorithm, restricted to partitions up to size 3, learns the target phonotactic patterns in all the case studies (see appendix). As mentioned, the algorithm considers partitions other than the ones discussed so far. For example, in each language, the pattern returned by algorithm with the partition given by the feature bundles [+high,+syllabic], [-high,+syllabic], and [-syllabic] shows that vowels are freely distributed with respect to [high]. Likewise, the feature [back] is freely distributed in every language considered here.

When the algorithm considered the partition that separated low vowels from non-low vowels from consonants, the ATR Contrastive and ATR Allophony languages showed free distribution. However, in the ATR Harmony language, the algorithm learned that if there is a third syllable following two syllables with low vowels, then the vowel in this third syllable cannot be low. This is a surface true fact about the sample in (1) and is likely due to the sparse input sample that was fed to the learner.

Note that because the algorithm considers all partitions by natural class, the algorithm does terminate eventually. Whether or not it does in a reasonable amount of time is another issue, given that there may be simply too many partitions by natural class.[7] Let us suppose that this is the case. How can the learner search this space to find the 'right' ones? This is one area where phonetic substance might bias learning (Wilson 2005) by requiring certain partitions to be ordered before others. Here however, I adopt the simple strategy of considering only the coarser partitions before finer ones.

[7] The number of partitions of a set of size n grows according to the Bell numbers, which is very (!) fast. Since we are considering partitions by natural class, the number of partitions will be much fewer, but may still grow too quickly. How quickly depends on the feature system.

On the other hand, it may be to the learner's advantage to never reach the finest partitions. It is possible that state-merging would yield little generalization when the learner uses the finest partition (where each segment forms its own natural class) because the neighborhoods are so diverse. Since the final grammar only accepts words that every acquired machine accepts, this potentially eliminates correct generalizations made earlier. Thus, if there are not too many partitions to exhaustively search, it may be necessary to stop the learner early.

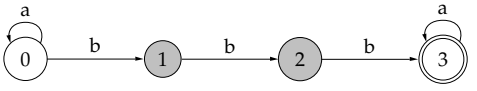
The question of which partitions the learner needs to consider will be answered when more complex phonotactic patterns are considered. For example, in Okpe, surface high vowels (which are realized as glides) do not participate in harmony though mid-vowels do, and low vowels partly participate (Hoffman 1973, Archangeli and Pulleyblank 1994).[8] For the learner to discover this pattern it will need to utilize a partition by natural class of size five to separate the consonants from the high vowels from the [+ATR] mid vowels from the [-ATR] mid vowels from the low vowels.

5. Neighborhood-Distinctness

In this section, I offer insight into why the algorithm is working, beginning with a definition of a relevant class of formal languages. A language is *neighborhood-distinct* iff there is a finite state acceptor for the language such that each state has its own unique neighborhood. Every phonotactic pattern considered to date is neighborhood-distinct (see also (Heinz 2006)). For example, the phonotactic patterns that constitute the cases studies here are all neighborhood-distinct, which can be easily verified upon inspection. This naturally leads to a non-trivial hypothesis that all phonotactic patterns are neighborhood-distinct.

This hypothesis is falsifiable since most formal languages are not neighborhood-distinct. One example is the language a^*bbba^*, a machine for which is shown in (12). It is provably not possible to construct an acceptor for this language because there will always be two states with the same neighborhoods.

(12)



Since the learner merges states with the same neighborhoods, it will not acquire this language, even upon exposure to large samples. In a sense, the learner fails here because it cannot distinguish 'three' from 'more than two'. In this way, neighborhood-distinctness is a property of patterns that learners can benefit from. Merging states with the same neighborhood helps ensure that neighborhood-distinct patterns are acquired.

Neighborhood-distinctness is a novel approach to locality in phonology. It provides a strategy for learning by limiting the kinds of generalizations that can be made. Thus it formalizes the "well-established generalization" about locality in phonology as a constraint on learning. Neighborhood-distinctness is also restrictive because finitely many languages are neighborhood-distinct (Heinz 2006).

6. Conclusions

There are many questions for future research. Are all phonotactic patterns neighborhood-distinct? I.e. will these results hold for more complex phonotactic patterns? Also, what kinds of patterns can the algorithm learn that are not considered possible and can they be eliminated by other factors? How can the algorithm be modified to handle noise (Angluin and Laird 1988)?

This paper presents a simple unsupervised batch learning algorithm that succeeds in three case studies. It generalizes correctly using only two notions, natural class and an abstract local notion of environment, the neighborhood. This leads to a novel, nontrivial hypothesis regarding phonotactic patterns: they are neighborhood-distinct.

Appendix: Results of other case studies

This appendix shows the learning input and the results of the algorithm for partitions π_0 and π_1.

[8] Thanks to Jie Zhang and Douglas Pulleyblank for bringing this case to my attention.

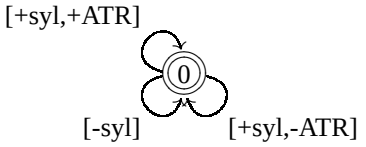
In the ATR Contrastive language, the sample shown in (13) was the input to the learner. Note that since the [ATR] feature is contrastive, its distribution is free, i.e. unpredictable.

(13)

1.	i	7.	bitki	13.	pAtkI	19.	bedkO	25.	ikOp
2.	ka	8.	montan	14.	pOtkO	20.	piptApu	26.	etIptUp
3.	eko	9.	I	15.	Ak	21.	mUtku	27.	potO
4.	puki	10.	kO	16.	IkU	22.	mitIpa	28.	kUtepA
5.	atapi	11.	kOn	17.	atEptAp	23.	AtetA	29.	kUptEpA
6.	kitepo	12.	IkUp	18.	dAkti	24.	pAki		

With the partition π_0, the learner successfully acquires the syllable structure phonotactic shown in (8). With partition π_1, the learner acquires the machine shown in (14).

(14)



This machine does not restrict the distribution of [ATR] at all.

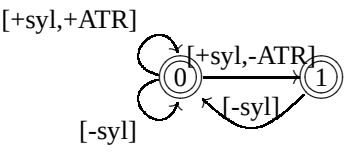
In the ATR Allophony language, [-ATR] vowels occur in closed syllables, and [+ATR] occur vowels elsewhere. The input sample for this language is shown in (15).

(15)

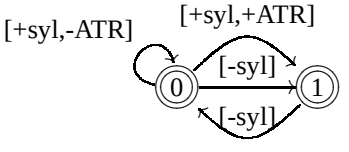
1.	a	5.	kitepo	9.	ko	13.	ateta	17.	pAtki
2.	i	6.	pati	10.	paki	14.	iku	18.	kUptapa
3.	ka	7.	atapi	11.	kutepa	15.	Ak	19.	pOtko
4.	puki	8.	eko	12.	poto	16.	kOn	20.	atEptAp
								21.	ikUp

As in the other case studies, the learner acquires the syllable structure phonotactic shown in (8) given partition π_0. With the partition π_1, the learner acquires the following machines in (16).

(16) a.



b.



References

Albright, Adam. 2006. Gradient Phonotactic effects: lexical? grammatical? both? neither? Talk handout from the 80th Annual LSA Meeting, Albuquerque, NM.

Albright, Adam and Bruce Hayes. 2002. Modeling English past tense intuitions with minimal generalization. *SIGPHON 6: Proceedings of the Sixth Meeting of the ACL Special Interest Group in Computational Phonology* :58–69.

Albright, Adam and Bruce Hayes. 2003. Rules vs. Analogy in English Past Tenses: A Computational/Experimental Study. *Cognition* 90:119–161.

Albro, Dan. 1998. Evaluation, implementation, and extension of Primitive Optimality Theory. Master's thesis, University of California, Los Angeles.

Albro, Dan. 2005. A Large-Scale, LPM-OT Analysis of Malagasy. Ph.D. thesis, University of California, Los Angeles.

Alderete, John, Adrian Brasoveanua, Nazarre Merchant, Alan Prince, and Bruce Tesar. 2005. Contrast analysis aids in the learning of phonological underlying forms. In *The Proceedings of WCCFL 24*. pages 34–42.

Angluin, Dana. 1982. Inference of Reversible Languages. *Journal for the Association of Computing Machinery* 29(3):741–765.

Angluin, Dana and Philip Laird. 1988. Learning from Noisy Examples. *Machine Learning* 2:343–370.

Archangeli, Diana. 1995. The Grounding Hypothesis and Javanese Vowels. In *Papers from the Fifth Annual Meeting of the Southeast Asian Linguistics Society (SEALSV)*, edited by S. Chelliah and W. de Reuse. pages 1–19.

Archangeli, Diana and Douglas Pulleyblank. 1994. *Grounded Phonology*. The MIT Press.

Baković, Eric. 2000. Harmony, Dominance and Control. Ph.D. thesis, Rutgers University.

Berko, Jean. 1958. The Child's Learning of English Morphology. *Word* 14:150–177.

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

Boersma, Paul. 2006. The Acquisition and Evolution of Faithfulness Rankings. In *Proceedings of the Eighth Meeting of the ACL Special Interest Group in Computational Phonology at HLT-NAACL*. New York City, USA.

Boersma, Paul and Bruce Hayes. 2001. Empirical tests of the Gradual Learning Algorithm. *Linguistic Inquiry* 32:45–86.

Chomsky, Noam and Morris Halle. 1968. *The Sound Pattern of English*. Harper & Row.

Coleman, John and Janet Pierrehumbert. 1997. Stochastic Phonological Grammars and Acceptability. In *Computational Phonology*. Somerset, NJ: Association for Computational Linguistics, pages 49–56. Third Meeting of the ACL Special Interest Group in Computational Phonology.

Eisner, Jason. 1997a. Efficient Generation in Primitive Optimality Theory. In *Proceedings of the 35th Annual ACL and 8th EACL*. Madrid, pages 313–320.

Eisner, Jason. 1997b. What constraints should OT allow? Talk handout, Linguistic Society of America, Chicago.

Eisner, Jason. 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)*, edited by Jason Eisner, Lauri Karttunen, and Alain Thériault. Luxembourg, pages 22–33.

Ellison, T.M. 1991. The Iterative Learning of Phonological Constraints. *Computational Linguistics* .

Frisch, S., J. Pierrehumbert, and M. Broe. 2004. Similarity Avoidance and the OCP. *Natural Language and Linguistic Theory* 22:179–228.

Frisch, Stephan. 1996. Similarity and Frequency in Phonology. Ph.D. thesis, Northwestern University.

Gildea, Daniel and Daniel Jurafsky. 1996. Learning Bias and Phonological-rule Induction. *Association for Computational Linguistics* .

Goldsmith, John. 2006. Information Thoery and Phonology. Slides presented at the 80th Annual LSA in Alburquerque, New Mexico.

Halle, Morris. 1978. Knowledge Unlearned and Untaught: What Speakers Know about the Sounds of Their Language. In *Linguistic Theory and Psychological Reality*. The MIT Prss.

Hayes, Bruce. 1999. Phonetically-Driven Phonology: The Role of Optimality Theory and Inductive Grounding. In *Functionalism and Formalism in Linguistics, Volume I: General Papers*. John Benjamins, Amsterdam, pages 243–285.

Hayes, Bruce. 2004. Phonological acquisition in Optimality Theory: the early stages. In *Fixing Priorities: Constraints in Phonological Acquisition*, edited by Rene Kager, Joe Pater, and Wim Zonneveld. Cambridge University Press.

Hayes, Bruce and Colin Wilson. 2006. The UCLA Phonotactic Learner. Talk handout from UCLA Phonology Seminar.

Heinz, Jeffrey. 2006. Learning Quantity Insensitive Stress Systems via Local Inference. In *Proceedings of the Eighth Meeting of the ACL Special Interest Group in Computational Phonology at HLT-NAACL*. pages 21–30. New York City, USA.

Hoffman, C. 1973. The Vowel Harmony System of the Okpe Monosyllabic Verb. In *Research Notes from the Department of Linguistics and Nigerian Languages*, volume 6. pages 79–111. University of Idaban.

Hopcroft, John, Rajeev Motwani, and Jeffrey Ullman. 2001. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Jarosz, Gaja. 2006. Richness of the Base and Probabilistic Unsupervised Learning in Optimality Theory. In *Proceedings of the Eighth Meeting of the ACL Special Interest Group in Computational Phonology at HLT-NAACL*. pages 50–59. New York City, USA.

Johnson, C. Douglas. 1972. *Formal Aspects of Phonological Description*. The Hague: Mouton.

Kaplan, Ronald and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20(3):331–378.

Karttunen, Lauri. 1998. The proper treatment of optimality theory in computational phonology. *Finite-state methods in natural language processing* :1–12.

Karttunen, Lauri. 2006. The Insufficiency of Paper-and-Pencil Linguistics: the Case of Finnish Prosody. Rutgers Optimality Archive #818-0406.

Kearns, Michael and Umesh Vazirani. 1994. *An Introduction to Computational Learning Theory*. MIT Press.

Kenstowicz, Michael. 1994. *Phonology in Generative Grammar*. Blackwell Publishers.

Ladefoged, Peter and Ian Maddieson. 1996. *Sounds of the World's Languages*. Blackwell Publishers.

Lin, Ying. 2002. Probably Approximately Correct Learning of Constraint Ranking. Master's thesis, University of California, Los Angeles.

Lodge, Ken. 1995. Kalenjin phonology and morphology: A further exemplification of underspecification and non-destructive phonology. *Lingua* 96:29–43.

McCarthy, John and Alan Prince. 1986. Prosodic Morphology. Ms., Department of Linguistics, University of Massachusetts, Amherst, and Program in Linguistics, Brandeis University, Waltham, Mass.

Merchant, Nazarre and Bruce Tesar. to appear. Learning underlying forms by searching restricted lexical subspaces. In *The Proceedings of CLS 41*. ROA-811.

Osherson, Daniel, Scott Weinstein, and Michael Stob. 1986. *Systems that Learn*. MIT Press, Cambridge, Massachusetts.

Pater, Joe. 2004. Exceptions and Optimality Theory: Typology and Learnability. Conference on Redefining Elicitation: Novel Data in Phonological Theory. New York University.

Pater, Joe and Anne Marie Tessier. 2003. Phonotactic Knowledge and the Acquisition of Alternations. In *Proceedings of the 15th International Congress on Phonetic Sciences, Barcelona*, edited by M.J. Solé, D. Recasens, and J. Romero. pages 1777–1180.

Prince, Alan and Paul Smolensky. 1993. Optimality Theory: Constraint Interaction in Generative Grammar. Technical Report 2, Rutgers University Center for Cognitive Science.

Prince, Alan and Paul Smolensky. 2004. *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell Publishing.

Prince, Alan and Bruce Tesar. 2004. Fixing priorities: constraints in phonological acquisition. In *Fixing Priorities: Constraints in Phonological Acquisition*. Cambridge: Cambridge University Cambridge: Cambride University Press.

Riggle, Jason. 2004. Generation, Recognition, and Learning in Finite State Optimality Theory. Ph.D. thesis, University of California, Los Angeles.

Riggle, Jason. 2006. Using Entropy to Learn OT Grammars From Surface Forms Alone. In *Proceedings of the 25th West Coast Conference of Formal Linguistics*, edited by Donald Baumer, David Montero, and Michael Scanlon. Cascadilla Proceedings Project.

Riggle, Jason. in preparation. Probably Approximately Correct Learning in Optimality Theory. November 2005 draft.

Stewart, J.M. 1967. Tongue root position in Akan vowel harmony. *Phonetica* 16:185–204.

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

Tesar, Bruce. 1998. An Interative Strategy for Language Learning. *Lingua* 104:131–145.

Tesar, Bruce and Paul Smolensky. 1998. Learnability in Optimality Theory. *Linguistic Inquiry* 29:229–268.

Tessier, Anne-Michelle. 2006. Stages of OT Phonological Acquisition and Error-Selective Learning. In *Proceedings of the 25th West Coast Conference of Formal Linguistics*, edited by Donald Baumer, David Montero, and Michael Scanlon. Cascadilla Proceedings Project.

Tucker, Archibald N. 1964. Kalenjin Phonetics. In *In Honour of Daniel Jones*, edited by D. Abercrombie, D. B. Fry, P. A. D. MacCarthy, N. C. Scott, and J. L. M. Trim. Longmans, London, pages 445–470.

Wilson, Colin. 2005. Learning Phonology With Substantive Bias: An Experimental and Computational Study of Velar Palatalization. In *UCLA Working Papers in Linguistics*, 11. Papers in Phonology 6.

Wilson, Colin. 2006. The Luce Choice Ranker. Talk handout, UCLA Phonology Seminar.

Proceedings of the 25th West Coast Conference on Formal Linguistics

edited by Donald Baumer,
David Montero, and Michael Scanlon

Cascadilla Proceedings Project Somerville, MA 2006

Copyright information

ISBN 1-57473-415-6 library binding

Ordering information

Orders for the library binding edition are handled by Cascadilla Press.
To place an order, go to www.lingref.com or contact:

Cascadilla Press, P.O. Box 440355, Somerville, MA 02144, USA
phone: 1-617-776-2370, fax: 1-617-776-2271, e-mail: sales@cascadilla.com

Web access and citation information

This entire proceedings can also be viewed on the web at www.lingref.com. Each paper has a unique document # which can be added to citations to facilitate access. The document # should not replace the full citation.

This paper can be cited as:

Heinz, Jeffrey. 2006. Learning Phonotactic Grammars from Surface Forms. In *Proceedings of the 25th West Coast Conference on Formal Linguistics*, ed. Donald Baumer, David Montero, and Michael Scanlon, 186-194. Somerville, MA: Cascadilla Proceedings Project.

or:

Heinz, Jeffrey. 2006. Learning Phonotactic Grammars from Surface Forms. In *Proceedings of the 25th West Coast Conference on Formal Linguistics*, ed. Donald Baumer, David Montero, and Michael Scanlon, 186-194. Somerville, MA: Cascadilla Proceedings Project. www.lingref.com, document #1448.