A formal analysis of Correspondence Theory

Amanda Payne, Mai Ha Vu, and Jeffrey Heinz
University of Delaware

1 Introduction

An influential version of Optimality Theory (OT) adopts Correspondence Theory (McCarthy & Prince, 1995), which explicitly invokes a correspondence relation between the elements in input strings and output strings. This representation allows Correspondence Theory to define a number of faithfulness constraints which help account for a variety of segmental and prosodic phenomena. Given that the parameters of Correspondence Theory are an integral part of OT, if Correspondence Theory is within a manageable level of computational complexity, it would lend credence to OT being a psychologically plausible phonological theory.

This paper provides a computational analysis of the complexity of GEN and Correspondence Theory. The computational complexity of the problems we study are stated in terms of the nature of the logic involved in stating their solutions. The two classes of logic this paper focuses on are Monadic Second Order (MSO) logic and First Order (FO) logic. We assume some familiarity with these logics and in particular the fact that MSO logic is strictly more powerful than FO logic. For example, it is known that MSO logic over a vocabulary for strings defines exactly the *regular* stringsets (Büchi, 1960). On the other hand, FO logic over the same vocabulary defines exactly the *star-free* stringsets (McNaughton & Papert, 1971). It is no accident that the star-free formal languages are a proper subset of the regular languages; the logic underlying the former is more powerful than the logic underlying the latter.

The first result of this analysis shows that the GEN *function*, which maps an arbitrary underlying form to an infinite set of input-output pairs, each of which stands in a correspondence relation, is not MSO-definable. One intepretation of this result is that it works against Correspondence Theory, since phonological constraints and maps have been argued to be within the regular boundary (Kaplan & Kay, 1994) and as mentioned MSO-definability is associated with this boundary. However, a more charitable interpretation is that GEN should be viewed as a relation, in which case its computational complexity is still open. We discuss this further in the relevant section below.

Second, we show that the set of input-output Correspondence-theoretic candidates from a *given* underlying representation is FO-definable. This result contrasts with the first one above, and highlights the importance of careful interpretation of complexity results.

Third, we present evidence from some case studies that the correct input-output Correspondence-theoretic candidate from a *given* underlying representation can be accomplished with FO-definable, language-specific, inviolable constraints without recourse to optimization. In fact, the case studies tentatively suggest that even weaker logics may be sufficient here, which is in-line with the research program that argues that phonological patterns are subregular (Heinz, 2010; Chandlee, 2014; Lai, 2015; Jardine, 2016). One way to interpret this result is that reliance on language-specific, inviolable constraints obviates the need for optimization without increasing computational complexity. However, we acknowledge that validity of this interpretation only extends insofar as the case studies are representative of phonological phenomena generally.

In sum, our work thus shows that the representations promoted in Correspondence Theory—that elements in the input string correspond to elements in the output string—are fairly restrictive computationally, provided that we do not use the GEN function itself in classic OT.

Our approach is similar in spirit to Potts & Pullum (2002), which uses logic to formalize constraints. The major difference is that our third result suggests that the complexity stays low even with language-specific, inviolable constraints, cf. (Scobbie et al., 1996; Jardine & Heinz, 2015). In this regard, our analysis suggests that there is merit to the representations invoked in Correspondence Theory beyond the role they play in Optimality Theory.

This paper is organized as follows. §2 presents the first two results that traditional Correspondence Theory's GEN function is not MSO-definable but that the set of candidates from a given input is FO-definable. In §3 we provide an FO-definable schema for phonological correspondences and illustrate it with some case studies meant to showcase its flexibility. §4 concludes with some additional discussion.

2 Correspondence Theory

Correspondence Theory (McCarthy & Prince, 1995) characterizes GEN as a function which maps an underlying form A to a set of ordered triples. Each triple is a candidate. The first element of the triple is the underlying form as a string. The second element is the surface form as a string. The third element is the correspondence relation which relates elements in the underlying string to elements in the surface string. An example is given in (1).

(1) Gen(/abc/) $\rightarrow \{$(/abc/, [abc], $\mathcal{R}_1$), (/abc/, [abcd], $\mathcal{R}_2$), (/abc/, [abcd], $\mathcal{R}_3$), $\dots\}$

Mathematically, each $\mathcal{R}$ is a subset of $N \times M$ where $N = \{1, 2, \dots n\}$ with n the length of the underlying string and $M = \{1, 2, \dots m\}$ with m the length of the surface string. In classic Optimality Theory, the set of candidates is infinite in size because there is no upper bound on the length of the output string. Of course, candidates whose output strings have many epenthesized segments are typically harmonically bounded.

2.1 *Representing Candidates* We model candidates with *relational structures*. This approach has its roots in finite model theory (Enderton, 2001; Hedman, 2004; Courcelle & Engelfriet, 2012). (See Potts & Pullum (2002) and Rogers et al. (2013) for model-theoretic approaches to phonology.) A relational structure contains a set of *domain elements* and a fixed number of relations over this set. To model Correspondence Theory, we only need unary and binary relations.

Unary relations indicate properties of the domain elements. The unary relations used here indicate whether the node belongs to the underlying string (*u*) or the surface string (*s*), and which phonological segment it is.

There are multiple ways to instantiate the last type of unary relation. One way is to have a unary relation for each symbol in the string. Thus, if *a* is a symbol in a string, there would be a unary relation *a* and elements in the domain could stand in this relation to indicate they represent such symbols. This is how Büchi (1960) modeled strings. However, another approach is phonologically motivated. We can let the unary relations stand for phonological features. In this approach, the unary relations indicate properties like *voiced, obstruent, coronal* and so on. In this paper, we avail ourselves of both types as needed. The primary reason for this is convenience, but it is also worth pointing out that predicates for features are FO-definable from segments and that predicates for segments are FO-definable from features.

Two binary relations are used here. One indicates whether two domain elements stand in the precedence relation ($<$). The other indicates whether two domain elements stand in the correspondence relation (c).

Relational structures can be displayed visually as graphs where the domain elements are nodes of the graph. These nodes are indexed with numbers to identify them. The unary relations each node belongs to is expressed by labeling the nodes according to the unary relation. Binary relations between elements in the structure are drawn as edges connecting nodes, and these edges are labeled to indicate which relation it represents.

We illustrate relational structures with three candidates (3)-(5) below. Candidate (3) shows a candidate that violates MAX. Candidate (4) shows a candidate that violates INTEGRITY, UNIFORMITY and IDENT. Candidate (5) shows a candidate that violates DEP and IDENT.

Note that in the interest of readability, instead of precedence, we use the edges in the graphs to show the successor relation. The successor relation is FO-definable from precedence as shown in (2).

(2) $(x \lhd y) = x < y \wedge \neg(\exists z)[x < z \wedge z < y]$

(3)

$$\begin{tikzcd}
a_u \; {\scriptstyle 1} \arrow[r, "\lhd"] \arrow[d, "c"] & b_u \; {\scriptstyle 2} \arrow[r, "\lhd"] \arrow[d, "c"] & c_u \; {\scriptstyle 3} \\
a_s \; {\scriptstyle 4} \arrow[r, "\lhd"] & b_s \; {\scriptstyle 5} &
\end{tikzcd}$$

(4)

$$\begin{tikzcd}
a_u \; {\scriptstyle 1} \arrow[r, "\lhd"] \arrow[d, "c"] & b_u \; {\scriptstyle 2} \arrow[r, "\lhd"] \arrow[d, "c"] \arrow[dr, "c"] & c_u \; {\scriptstyle 3} \arrow[d, "c"] \\
a_s \; {\scriptstyle 4} \arrow[r, "\lhd"] & b_s \; {\scriptstyle 5} \arrow[r, "\lhd"] & c_s \; {\scriptstyle 6}
\end{tikzcd}$$

(5)

$$\begin{tikzcd}
a_u \; {\scriptstyle 1} \arrow[r, "\lhd"] \arrow[d, "c"] & b_u \; {\scriptstyle 2} \arrow[dr, "c"] & \\
a_s \; {\scriptstyle 3} \arrow[r, "\lhd"] & b_s \; {\scriptstyle 4} \arrow[r, "\lhd"] & c_s \; {\scriptstyle 5}
\end{tikzcd}$$

2.2 *The GEN function* The GEN function is *not* MSO-definable. The core reason is stated in Courcelle & Engelfriet (2012) as Fact 1.37, where U is an arbitrary input string and S is a set of candidates: "For every monadic second-order transduction f there exists an integer k such that, if f transforms a relational structure U into a relational structure S, then $|S| \leq k|U|$."

In other words, the size of the relational structure must be bounded by some integer $k \times n$, where n is the size of the input and k is fixed in advance. Here we do not explicitly provide a relational structure for a *set* of candidates, but we do not need to. Since there is no upper bound on the length of the surface string in a candidate, there will always be some candidate larger than $k \times n$ and hence any set containing it will also be too large. Consequently, the output of GEN must be finite in size for it to be an MSO definable function. Furthermore, since the set of candidates output by GEN in classic OT is generally assumed to be infinite in size, it cannot be described with any MSO-definable *function*.

What does this result mean for phonological theory? There are three possibilities.

The first is that instead of analyzing GEN as a function, we should analyze it as a relation. We are unaware of a model-theoretic or logical treatment of MSO-definable relations. Although GEN is not a MSO-definable function, it *may* be a MSO-definable *relation*. One way to approach this from a model theoretic perspective may be with infinite structures (Benedikt et al., 2001).

The second approach is to reconceive GEN as a function from underlying strings to candidates to a function from underlying strings to *contenders*. The contenders are the non-harmonically bounded candidates. Riggle (2004) shows that the contenders typically form a finite set. So it may be the case that GEN as a function from underlying strings to contenders is MSO-definable.

The third approach is to take this result as a reason to reject classic OT in favor of Harmonic Serialism (McCarthy, 2008) where GEN is a function which maps an underlying string w to a set of candidates, each of which differs at most minimally from w. Restricting GEN in this way might also make it an MSO-definable function.

To summarize, it is interesting to observe that GEN is not a MSO-definable function, and there are several alternatives worth pursuing.

2.3 *The set of candidates for a given input* If we fix one particular input string, then for this input, the output candidates can be defined with FO-logic. Only five logical statements are needed to describe the infinite set of possible candidates for a given input. The logical statements, listed in (6)-(10) are all described with FO-logic over the relational structures described earlier. These statements are explained further below.

(6) Precedence can only be between nodes on same tier:
$(\forall x, y)\Big[x < y \rightarrow [u(x) \wedge u(y)] \vee [s(x) \wedge s(y)]\Big]$

(7) Correspondence only between different tiers:
$(\forall x, y)\big[\mathtt{c}(x, y) \rightarrow (u(x) \wedge s(y))\big]$

(8) All nodes are either on UR or SR:
$(\forall x)[u(x) \vee s(x) \wedge \neg[u(x) \wedge s(x)]]$

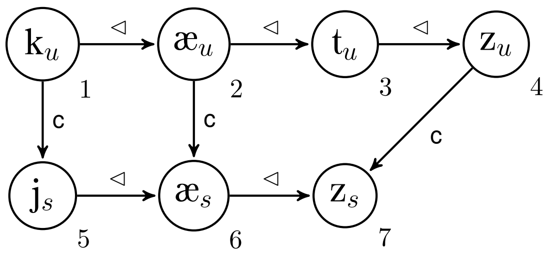
(9) SR nodes form a string:
$(\forall x, y)[[s(x) \wedge s(y)] \rightarrow [x < y \vee y < x]$

(10) UR is /kætz/:
$(\exists v, x, y, z)[v < x \wedge v < y \wedge v < z \wedge x < y \wedge x < z \wedge y < z \wedge u(v) \wedge u(x) \wedge u(y) \wedge u(z) \wedge k(v) \wedge$ æ(x) $\wedge t(y) \wedge z(z)]$

Statement (6) says that precedence relationships can only be between nodes that belong to the same 'tier', i.e. there cannot be precedence between an underlying and a surface node. Conversely, correspondence edges can only be between nodes that are on different tiers (7). All nodes have to be either belong to the UR or the SR, so they cannot be both or neither (8). The surface tier has to form a string, where the definition of a string is a set of nodes where the precedence relation establishes a linear order (9). Lastly, we need a fixed input string and statement (10) provides this. In this example, the input string is the word /kætz/, and it is defined segment by segment.

The graph in (11) shows an example of an output candidate which satisfies these five constraints. Again, for readability, we show the graph with successor relations instead of precedence relations. (Note that this is not the correct candidate, as we would expect [kæts] to surface, and not [jæz].)

(11)



Every relational structure which satisfies the five statements is a valid candidate for the underlying string /kætz/.

Generally, for any given underlying string $w = a_1 a_2 \ldots a_n$ the fifth statement can be written as follows.

(12) $(\exists x_1, x_2 \ldots x_n)\Big[\bigwedge(x_i < x_j)_{i<j} \bigwedge(a_i(x_i))_{1\leq i\leq n} \bigwedge(u(x_i))_{1\leq i\leq n}\Big]$

In (12), $\bigwedge(x_i < x_j)_{1\leq i<j\leq n}$ stands for the conjunction of statements $x_i < x_j$ whenever i and j take on values inclusively between 1 and n and $i < j$. Similarly, $\bigwedge(a_i(x_i))_{1\leq i\leq n}$ means a conjunction of n statements; for each i between 1 and n inclusive, the ith statement says x_i stands in the unary relation a_i. Similarly, $\bigwedge(u(x_i))_{1\leq i\leq n}$ means each of variable stands in the unary relation which indicates it is an underlying element.

In order to get the right mapping of /kætz/ to [kæts], we need additional, language specific constraints. Examples for dealing with simple phonological mappings, such as assimilation, deletion, and insertion are shown in the next section.

3 Phonological maps using FO-constraints and Correspondence Theory

3.1 *Assimilation* We show how assimilation can be accounted for with the example of English obstruent devoicing. The generalization is that voiced obstruents become voiceless after voiceless sounds. The desired output for the input /kætz/ is shown in (13) – this would be the same as the optimal candidate picked by OT constraints.

(13)

$$\begin{tikzcd}
\text{k}_u \ {}_{1} \arrow[r, "\lhd"] \arrow[d, "\mathsf{c}"] & \text{æ}_u \ {}_{2} \arrow[r, "\lhd"] \arrow[d, "\mathsf{c}"] & \text{t}_u \ {}_{3} \arrow[r, "\lhd"] \arrow[d, "\mathsf{c}"] & \text{z}_u \ {}_{4} \arrow[d, "\mathsf{c}"] \\
\text{k}_s \ {}_{5} \arrow[r, "\lhd"] & \text{æ}_s \ {}_{6} \arrow[r, "\lhd"] & \text{t}_s \ {}_{7} \arrow[r, "\lhd"] & \text{s}_s \ {}_{8}
\end{tikzcd}$$

In order to state the constraints in FO-logic, we define two predicates: (14) defines a segment where a voiceless sound is followed by a voiced obstruent, and (15) defines a segment where a voiceless sound is followed by a voiceless obstruent within a string.

(14) A voiceless sound followed by a voiced obstruent:
$TD(x,y) \overset{\text{def}}{\equiv} x \lhd y \wedge voiceless(x) \wedge voiced(y) \wedge obstruent(y)$

(15) A voiceless sound followed by a voiceless obstruent:
$TT(x,y) \overset{\text{def}}{\equiv} x \lhd y \wedge voiceless(x) \wedge voiceless(y) \wedge obstruent(y)$

The devoicing process can be stated in part with a FO-definable constraint using implication (16): if the underlying segment is a voiceless sound followed by a voiceless obstruent, than they have to be in correspondence with a voiceless segment and voiceless obstruent, respectively.

(16) $(\forall x_1, y_1, x_2, y_2)[(TD(x_1, y_1) \wedge u(x_1) \wedge \mathsf{c}(x_1, x_2)) \rightarrow TT(x_2, y_2)]$

For the devoicing example in (13), the variables in constraint (16) are satisfied when $x_1 = 3$, $x_2 = 7$, $y_1 = 4$, and $y_2 = 8$.

As in OT, additional constraints are needed to ensure other candidates are disallowed. For example, constraints similar to UNIFORMITY (17), INTEGRITY (18), MAX (19), and DEP (20) are shown below.

(17) $(\forall x, y, z)\big[(u(x) \wedge \mathsf{c}(x,y) \wedge \mathsf{c}(x,z)) \rightarrow y = z\big]$

(18) $(\forall x, y, z)\big[(s(y) \wedge \mathsf{c}(x,y) \wedge \mathsf{c}(z,y)) \rightarrow x = z\big]$

(19) $(\forall x)(\exists y)\big[u(x) \rightarrow \mathsf{c}(x,y)\big]$

(20) $(\forall x)(\exists y)\big[s(x) \rightarrow \mathsf{c}(y,x)\big]$

We additionally need constraints like IDENT(F) for each feature F except for the features *voiced* and *voiceless*. The general form of the constraint for the features that are faithful everywhere is shown in (21).

(21) $(\forall x, y)\big[(u(x) \wedge F(x) \wedge \mathsf{c}(x,y)) \rightarrow F(y)\big]$

For example, the feature *coronal* is faithful everywhere. The structure in (13) above satisfies (21). There are two relevant cases when $x = 3$ and $y = 7$ and when $x = 4$ and $y = 8$. In both instances the y element is coronal as (21) demands.

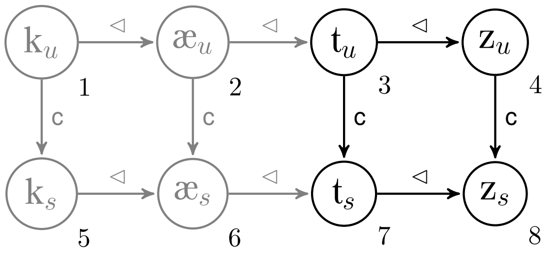
For the features *voiced* and *voiceless*, they must be faithful everywhere except the devoicing environment identified above. In OT, this is accomplished with language-specific constraint rankings, but in the declarative approach here, such environments are specified in language-specific constraints.

(22) $(\forall x, y)\Big[\big(u(x) \wedge voiced(x) \wedge \mathtt{c}(x,y) \wedge \neg(\exists z)[TD(z,x)]\big) \rightarrow voiced(y)\Big]$

The language-specific, inviolable constraint (22) completes the analysis of the devoicing.

It might be possible to get the same desired candidate as a conjunction of banned substructures similar to Jardine's (2016) analysis of tonal association. If so, it would indicate that the logical power needed to state the constraints is actually more restrictive than FO-logic. For instance, the constraint in (23) would be just one of several more statements that forbid different types of substructures. The idea is that the candidate whose surface string is (13) is the *only* candidate that satisfies them all (that is, it does not contain any forbidden substructure). We illustrate the statement with a graph, where the highlighted part shows the banned substructure.

(23) $\neg(\exists x, y)[TD(x) \wedge u(x) \wedge \mathtt{c}(x,y) \wedge TD(y)]$



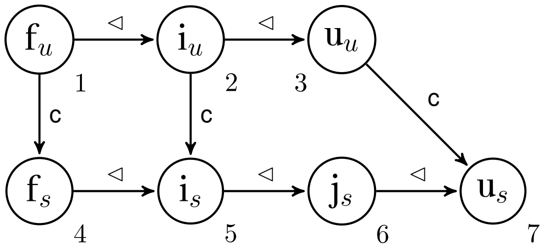
It is an open question whether assimilation and other phonological processes can be approached in this way. In the interest of simplicity, we only give the conditional-style constraints for the remainder of the case studies.

3.2 *Insertion* One example for insertion is Hungarian [j]-insertion in vowel-hiatus (Siptar, 2005). Again, we define the condition for insertion in (24): a vowel that is succeeded by another vowel. Then the constraint in (25) simply states that when there is a VV sequence in the underlying string, there must be a [j] element in the surface string that intervenes between the correspondents of the two underlying vowels. This is also true the other way around: if there is a [j] on the surface without a corresponding underlying element, then it must be the case that underlyingly, there was vowel hiatus. The correct candidate is represented in (26) for the input [fiu].

(24) $VV(x) \overset{\text{def}}{\equiv} (\exists y)[x \lhd y \wedge vocalic(x) \wedge vocalic(y)]$

(25) $(\forall x)\Big[(VV(x) \wedge u(x)) \leftrightarrow (\exists y, z)[\mathtt{c}(x,y) \wedge y \lhd z \wedge j(z) \wedge \neg(\exists w)[\mathtt{c}(w,z)]]\Big]$

(26)



Additional constraints are necessary to ensure the rest of the surface string is faithful. In particular constraints (17)-(19) and (21). Observe that the constraint (25) replaces (20), which banned epenthesis. Here epenthesis is only permitted in the particular environment specified by (25).

3.3 *Deletion* Our example for deletion is Tibetan consonant deletion in consonant clusters (Halle & Clements, 1983). The generalization depends on syllable structure, as consonant deletion only applies in onset clusters. We simplify onset clusters here as word initial consonant clusters.

Again, we define word initial consonant clusters very similarly to vowel hiatus: two successive elements that stand in the *consonantal* relation, and are not preceded by any other element (27). Then we give a constraint, stated as a biconditional: if there is a consonant cluster in the underlying string then there is no element in the surface string which corresponds to the first underlying consonant (28). The converse is also true: if there is an element in the underlying form that has no correspondent in the surface, it must be the case that the underlying element is the first consonant in the cluster. (29) shows the correct candidate.

(27) $\#CC(x) \overset{\text{def}}{\equiv} (\exists y)\big[x \lhd y \wedge consonant(x) \wedge consonant(y) \wedge \neg(\exists z)[z \lhd x]\big]$

(28) $(\forall x)\Big[\#CC(x) \wedge u(x) \leftrightarrow \neg(\exists y)[\mathsf{c}(x,y)]\Big]$

(29)

$$\begin{tikzcd}
\text{b}_u\ 1 \arrow[r, "\lhd"] & \text{dʒ}_u\ 2 \arrow[r, "\lhd"] \arrow[d, "\mathsf{c}"] & \text{u}_u\ 3 \arrow[r, "\lhd"] \arrow[d, "\mathsf{c}"] & \text{b}_u\ 4 \arrow[r, "\lhd"] \arrow[d, "\mathsf{c}"] & \text{ʃ}_u\ 5 \arrow[r, "\lhd"] \arrow[d, "\mathsf{c}"] & \text{i}_u\ 6 \arrow[d, "\mathsf{c}"] \\
 & \text{dʒ}_s\ 7 \arrow[r, "\lhd"] & \text{u}_s\ 8 \arrow[r, "\lhd"] & \text{b}_s\ 9 \arrow[r, "\lhd"] & \text{ʃ}_s\ 10 \arrow[r, "\lhd"] & \text{i}_s\ 11
\end{tikzcd}$$

As before, the candidate with the correct surface form has to satisfy other faithfulness constraints given by (17), (18), (20), and (21). (19), which states the MAX constraint, is replaced by (28), which only allows deletion in the defined environment.

4 Conclusion

This paper provides an analysis of the role of Correspondence Theory in OT and phonological theory more generally. The analysis is not complete in the sense that there are still unanswered questions worth pursuing.

For instance, it was shown that the GEN function is *not* MSO-definable. This result means that researchers should study the MSO-definability of GEN as a relation, GEN as a function which maps underlying strings to contenders, and GEN as it is conceived in Harmonic Serialism.

On the other hand, it was also shown that the infinite set of candidates for a given input is MSO-definable, in fact FO-definable. This lends some credence to the plausibility of correspondence theory and classic OT in general.

We also showed examples of common processes in phonology that are amenable to a Correspondence Theoretic analysis with FO logic. These analyses do not rely on optimization but are declarative in the sense that every constraint is inviolable. Future work would examine long-distance phonological processes such as dissimilation, vowel harmony, and consonantal harmony. Finally, it is of interest to know the extent to which phonological maps can be expressed with more restrictive logics (such as the conjunction of forbidden substructures) using Correspondence-theoretic representations.

References

Benedikt, Michael, Leonid Libkin, Thomas Schwentick & Luc Segoufin (2001). A model-theoretic approach to regular string relations. *Logic in Computer Science, Symposium on* 0, p. 0431.

Büchi, J. Richard (1960). Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly* 6:1-6, 66–92.

Chandlee, Jane (2014). *Strictly local phonological processes*. Ph.D. thesis, University of Delaware.

Courcelle, Bruno & Joost Engelfriet (2012). *Graph structure and monadic second-order logic: a language-theoretic approach*, vol. 138. Cambridge University Press.

Enderton, Herbert B. (2001). *A Mathematical Introduction to Logic*. Academic Press, 2nd edn.

Halle, Morris & George Clements (1983). *Problem book in phonology: a workbook for introductory courses in linguistics and in modern phonology*. MIT Press.

Hedman, Shawn (2004). *A First Course in Logic*. Oxford University Press.

Heinz, Jeffrey (2010). Learning long-distance phonotactics. *Linguistic Inquiry* 41, 623–661.

Jardine, Adam (2016). *Locality and non-linear representations in tonal phonology*. Ph.D. thesis, University of Delaware.

Jardine, Adam & Jeffrey Heinz (2015). Markedness constraints are negative: an autosegmental constraint definition language. *Chicago Linguistic Society*.

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

Lai, Regine (2015). Learnable vs. unlearnable harmony patterns. *Linguistic Inquiry* 46:3, 425–451.

McCarthy, John J. (2008). The gradual path to cluster simplification. *Phonology* 25:2, 271–319.

McCarthy, John J & Alan Prince (1995). Faithfulness and reduplicative identity .

McNaughton, Robert & Seymour Papert (1971). *Counter-Free Automata*. MIT Press.

Potts, Christopher & Geoffrey K Pullum (2002). Model theory and the content of ot constraints. *Phonology* 19:03, 361–393.

Riggle, Jason Alan (2004). *Generation, recognition, and learning in finite state Optimality Theory*. Ph.D. thesis, UCLA.

Rogers, James, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert & Sean Wibel (2013). Cognitive and sub-regular complexity. Morrill, Glyn & Mark-Jan Nederhof (eds.), *Formal Grammar*, Springer, vol. 8036 of *Lecture Notes in Computer Science*, 90–108.

Scobbie, James M., John S. Coleman & Steven Bird (1996). Key aspects of declarative phonology. Durand, Jacques & Bernard Laks (eds.), *Current Trends in Phonology: Models and Methods*, European Studies Research Institute, Manchester, UK, vol. 2, 685–709. University of Salford.

Siptar, Peter (2005). The phonology of Hungarian vowel clusters. *Magyar Nyelv (Hungarian Language)* 3:101, 282–304.