Recursive Join Learning using Candidate Maps[1]

Nazarré Merchant, Eckerd College

3/28/2015

Section 1. Introduction.

Learning a grammar in Harmonic Serialism (Prince and Smolensky 1993/2004) requires the learner to determine a ranking that maps observed forms to themselves and non-faithfully mapped forms non-faithfully. The fact that observed forms map to themselves is an immediate consequence of the structure of HS. In HS, an observed output form is produced from a derivation that is composed of a sequence of maps, $A_1 \rightarrow \ldots \rightarrow A_{n-1} \rightarrow A_n$. By definition, for a sequence to be convergent, the final output form, A_n, must be identical to the previous form in the sequence, so that $A_{n-1} = A_n$. This ensures that chain shifts[2] cannot obtain in HS regardless of constraint or operation constituency (McCarthy (2000) notes HS's inability to produce chain shifts for certain sets of constraints).

Problematic from the learner's perspective is that not all input-output mappings are given – a learner may know that an output form is not in the surface repertory, but may not know what it maps to, whether it is a faithful form, like all observed forms, or which non-faithful form it maps to. This ambiguity can obscure target-language ranking information from the learner.

Further compounding the learning task is that there is often a glut of non-surface present forms, and the learner must sort through these to find informative maps, faithful or not. Tessier and Jesney (2014) make an important observation, identifying an HS system in which exactly this happens – the learner must find a certain non-faithfully mapped form, determine what it maps to, and extract ranking information from this map. The core of the matter, as they clearly lay out, can be stated in two questions: (1) which non-surface present underlying forms does one attempt to learn from, and (2) given such a form with an unknown output, what ranking information can be determined of the target language?

This paper proposes approaching these question using a learning algorithm, Recursive Join Learning (RJL), described below, that relies on two ideas utilizable in every Harmonic Serialism system. The first is that every HS system inherently organizes the forms of its system based on a precise notion of similarity, built from the idea that two forms are one unit away from each other if there is an operation that transforms one into the other.

Notions of similarity pervade discussions of linguistic forms; for example, a five-syllable form that is parsed into left-aligned trochaic feet with the final syllable parsed as a foot,

[1] Thanks to Alan Prince for discussion and a close read-through of the paper.

[2] A chain shift occurs when /A/ → [B] and /B/ → [C], where [B] and [C] represent the final step of an HS derivation.

(**σσ**)(**σσ**)(**σ**), may be thought to be more similar to a form that leaves the final syllable unparsed, (**σσ**)(**σσ**)σ, than to a right-aligned iambic parse which leaves the first syllable unparsed, σ(σ**σ**)(σ**σ**), depending on one's notion of similarity.

A notion of similarity in HS can be made sufficiently precise, so that one can give a *distance* between two forms of the typology, whether they appear in the same language or not. This imposes a *metrical* structure on the forms of the typology – the forms form a *metric space*. Having a metric space on the forms means there are rich structural relations amongst the forms of the system, relations exploitable for numerous purposes – here we show how they can be used to build a powerful learning tool.

In the example of metrical feet, given suitable operations, the distance between (**σσ**)(**σσ**)(**σ**) and (**σσ**)(**σσ**)σ is 1, while between (**σσ**)(**σσ**)(**σ**) and σ(σ**σ**)(σ**σ**) is 5. Note that this distance, and its derived notion of similarity, is different from a constraint-based OT notion of similarity. For example, presuming All-feet-left (AFL) and All-feet-right (AFR) constraints, (**σσ**)(**σσ**)(**σ**) is *right*-aligned, surfacing only when AFR ≫ AFL, as is σ(σ**σ**)(σ**σ**), while (**σσ**)(**σσ**)σ is left-aligned.

A terminological note: throughout I use the term *form* to mean any object that can serve as either an input to Gen or as an output of Gen, for a given HS system. Since HS requires that outputs of Gen serve as possible inputs to Gen this conflation of categories is, at times, needed. A learner then is presented with a set of forms, which are the outputs of a set of derivations, and learning is determining which forms map to which and what ranking produces these mappings.

Returning to the notion of similarity, the distance metric presented here will be used to produce what I am calling a *Candidate Map* (CM) of the typology, built from *Local Candidate Maps* (LCM) centered on the forms of the system. An LCM centered on a form is a graph of all the forms of the typology that are zero and one unit away from the given form, organized by distance, so that a form, *f*, is connected by an edge to all forms, *g*, that are one unit away from it. This graph structure is then augmented with a map structure – arrows between connected forms representing which markedness constraints prefer which forms. This structure organizes for the learner the forms that should be considered as inputs – which ones to consider for learning purposes – those that are closest to the faithfully mapped forms are investigated first, proceeding away using the map. The labeled arrows between forms allow the learner to determine ranking information entailed by a map that traverses that link. The Candidate Map of the typology is then the graph union of all of the local candidate maps – the graph union being the graph that contains all the nodes and edges of the constituent unioned graphs.[3]

[3] http://en.wikipedia.org/wiki/Graph_operations#Binary_operations

The second idea presented here, used in the learning algorithm, addresses the second question introduced above: a learner may only know that a form is not surface present, and therefore not necessarily faithfully mapped; how can useful ranking information be determined in this case? The answer lies in first making an assumption about the organization of the lexicon, namely, those forms that are closest to the observed forms are the least likely to be faithfully mapped, embodying the notion of contrast maximization. This allows the learner to identify forms that are likely not faithfully mapped. The learner can proceed assuming that indeed they are non-faithful maps by marking all information gleaned as contingent, readily expelled when contradiction appears. Other techniques, not explored here, obtain: weighting the information based on a given likelihood function, for example.

Given non-faithfully mapped forms, there is still the question of how to determine ranking information. The answer to this question lies in the logic of candidate selection in a single step of an HS derivation. Note that the candidate selection at the step level is exactly Optimality Theory candidate selection, and hence the logic of OT bears heavily on licit HS maneuvers, since the two, viewed sufficiently closely, are identical.

Each possible non-faithful map for an input form, *f*, when compared to the faithful map, gives rise to an ERC (Prince 2002), a vector whose entries come from the set {W, e, L}. Each ERC delimits exactly the set of total orders which select the desired winning candidate over the desired losing candidate, here the faithful candidate. So the set of non-faithfully mapped forms gives rise to a set of ERCs, one for each form. The learner knows that the target language must come from the union of the total orders delimited by these ERCs. This union may be an unwieldy object, not practically portable for learning purposes. Here I propose using the join of ERCs (Merchant 2008, Merchant 2011) which, in this case, yields a single ERC that represents the smallest ERC grammar that contains all the total orders of the non-faithful maps.

Roughly, the join of two ERCs is an easily computed ERC that captures shared ranking information. It does this by identifying which constraints must be dominated in a grammar that contains the total orders consistent with the two given ERCs. Those constraints that have an L in both ERCs must be dominated in their shared ranking information. This is represented by an L in the join. Potential dominators of these L's are those constraints that have a W in either of the joinards. The join places a W in these constraints. Non-domination crucial e's are placed elsewhere. This construction, which is the join, surprisingly has the property of being the smallest grammar that contains the two, and represents their shared ranking information (this follows immediately from Merchant 2008:111).

These two ideas, the Candidate Map and the Join, then can be amalgamated into a learning algorithm that provides insights into the important learning problem posed by

Tessier and Jesney, and solves the particular instance that they present – how do we get useful non-faithful mapping information from only faithful maps?

As noted by Tessier and Jesney, HS requires a step away from the identity map (Prince & Tesar (2004), Hayes (2004), Smolensky (1996):729[4]) in which useful ranking information can be determined by assuming the fixed-point property, /A/→[A], (note: one must also step away in OT). The candidate map and the join together, in the algorithm below, give useful intermediate forms that are learning exploitable.

The RJL algorithm presumes that the learner is presented with the overt forms of the target language. Learning then proceeds by determining ranking information from the identity map. Overt forms in HS must map to themselves and so the learner can assume safely that the faithful candidate wins (this is precisely the identity map learning of Prince and Tesar 2004). Ranking requirements codified in ERCs are then stored from the faithful candidate beating each of the non-faithful candidates. After identity map learning, the learner iteratively applies RJL by selecting successively those forms that are one unit distant from the faithfully mapped inputs. For each of these forms, the set of ERCs produced by the non-faithful maps are joined together (each is first checked for consistency with the data store of ERCs). This joined ERC is added to the data store of ERCs that must be satisfied by the target language. This process is repeated with forms two units away, then three, repeating until a distance, k, is reached. Termination at distance k is obtained when all the peripheral, k-distant forms, either (1) are one unit away from forms already encountered by the algorithm, or (2) produce no new ranking information. This second condition is needed because, for example, in a system having certain types of deletion, there is an infinite set of non-informative mappings. For example, for a language that deletes final codas, a valid input to [CV] is /CVC/, which has a valid input of /CVCC/, which has a valid input of /CVCCC/, and so on. The map CVCCC → CVCC is no more informative than CVC → CV, and since CVC → CV is encountered before the former, and its ranking information determined, the more distant points can be omitted from the search.

The forms of the system demonstrated here are finite, and hence the algorithm is guaranteed to terminate, as it will with all systems that have a finite number of forms. It is also guaranteed to terminate on all languages of systems that have an infinite number of forms. This is true because regardless of the number of forms in the system, there is only a finite amount of ranking information about any language in any HS system. The expanding shells around the fixed points cannot each determine some new ranking fact, and so the algorithm will always terminate.

[4] It is interesting to note that Smolenksy does not articulate the identity map assumption as typically construed, but that his lexical selection algorithm reduces exactly to it when the OT system it operates on has only markedness and faithfulness constraints.

A rough pseudo-code outline of the algorithm follows and is expanded upon in later sections.

(1) Pseudo-code of Recursive Join Learning
  (a) Apply Identity Map learning to the overt forms of the language. Add ERCs produced to a data store of ERCs.
    a. So, for [A], produce G_0:/A/→[A], and concomitant ERCs
  (b) For each overt form, A, create a local candidate map centered on each form that is one unit away from A and that is not faithfully mapped.
    For each of these local candidate maps, get ranking information by joining the ERCs produced by comparing non-faithful maps to the faithful.
    a. So, for B s.t. |B−A|=1, set $G_n = G_{n-1} \cup$ G:/B/ → [A], produce ERCs
  (c) Repeat step (b) until all forms from the previous step (b) either are one unit away only from forms previously encountered or did not produce any non-entailed ERCs.

The paper is organized thusly: the next section contains a single HS system of interest. Section 3 defines Candidate Maps. Section 4 articulates the join. Section 5 produces the Recursive Join Learning with Candidate Maps algorithm. The final section concludes the paper.

Section 2. Bantu-like System

To concretize the discussion, I present a single HS system manifesting the relevant learning complexities. The system is a modified and simplified version of the HS system presented in Tessier and Jesney, which they use to model Punu, a Bantu language spoken in Gabon and the Republic of Congo (Kwenzi Mikala 1980 and Hyman 2002 and 2008). Their interest in Punu stems from assimilation processes and positional restrictions it exhibits in final position (see Bennett 2013 for an analysis of Bantu language assimilation and dissimilation). In their Punu-like language these final restrictions are driven by rankings between two markedness constraints. As they show, this crucial ranking cannot be discerned from the Identity Map. The system presented here, as we will see, shares this crucial property of having non-ID map discernable rankings, but is simplified to sharpen focus on the relevant ideas.

In this system, which I call Bantu-Lite (BL), each form of the system (forms being both the inputs and outputs) consists of exactly one segment. Each segment is completely specified by two binary features: ±ATR and ±Low. This means there are only four inputs to the system and four possible outputs. Despite its simplicity, the system fully instantiates the problem diagnosed by Tessier and Jesney. To increase perceptibility, each form of the system is represented by a single letter, either ‘a’, ‘A’, ‘h’, or ‘H’. The ATR feature is encoded by capitalization: capital letters correspond to +ATR (‘A’ and ‘H’),

while lower-case corresponds to –ATR ('a' and 'h'). The Low feature is encoded lexicographically: +Low is represented with 'a' or 'A', while –Low is represented with 'h' or 'H'.

(2) Forms of Bantu-Lite

Input:	*IN* = {a, A, h, H}
Outputs:	*OUT* =*IN* = {a, A, h, H}

There are five constraints in BL, listed in (3).

(3) Constraints of BL

*[+Low]	violated once by 'a' and 'A'
*[–ATR, –Low]	violated once by 'h'
Have[–ATR, –Low]	violated once by 'a', 'A', and 'H'
Id[Low]	violated by candidates whose outputs differ in Low specification from their inputs
Id[ATR]	violated by candidates whose outputs differ in ATR specification from their inputs

These are the same as the constraints of T&J except that their non-final constraint has been replaced with Have[–ATR, –Low]. This is done, not because of linguistic motivations but because of expositional ones.[5] Replacing their non-final constraint allows this system to ignore multi-syllabic inputs, reducing the number of inputs and outputs considered. The T&J system is composed of multi-syllabic inputs where each syllable is uniquely determined by the nucleus segment; onsets, codas, and other syllable distinguishing properties are ignored. The nucleus is completely specified by the values of two binary features, Low and ATR. In the system presented here, BL, inputs are the same as in T&J, only restricted to one syllable inputs.

Specification of an HS system requires explicit definition of its operations – implicit definitions will not do.[6] Here there are two operations. Each one flips the value of one feature in the segment.

[5] Even so, constraints of the form, 'Have something', abound. Cf. Onset. Furthermore, paired markedness constraints, one preferring one set of configurations, the other a complementary set, exist in various permutations. E.g. Alignment constraints over forms of relevant sizes exhibit this property, like Main Left and Main Right constraints (stress on leftmost/rightmost syllable) on two syllable forms.

[6] Much hay has been made from the claim that HS differs from OT in that HS operations 'do one thing', while OT Gen does more. Often unnoted is that operations are, by definition, one thing, even if that thing is quite complex. For example, an unremarked upon operation might be one that builds a foot – which involves assignment of headedness, connection of association lines, boundary delimitation, prosodic word integration, amongst other things imaginable. What constitutes 'one thing' remains a central issue in studies of Harmonic Serialism.

(4) Operations:

Flip ATR	+ATR → –ATR	A → a, H → h
	–ATR → +ATR	a → A, h → H
Flip Low	+Low → –Low	a → h, A → H
	–Low → +Low	h → a, H → A

The factorial typology of BL contains eleven languages (determined using OTWorkplace (Prince, Tesar, & Merchant (2014)). It is worth noting that there are (at least) two sensible definitions of what constitutes a language and what constitutes a grammar in Harmonic Serialism. Here we define a *language* to be the set of derivations produced by a single total order, over the inputs of the system. A *grammar* for this language is then defined to be the set of total orders that produces exactly these derivations.

This definition contrasts with an alternative definition, not used here, in which a language is a set of inputs and corresponding outputs, produced by a single total order; and a grammar for that language is the set of total orders that map each input to its corresponding output. Under this definition of grammar, two total orders can belong to the same grammar but produce different sets of derivations – as long as they have the same set of input/output pairs. A crucial distinguishing difference between the two definitions is that the former always permits the grammars of the languages to be characterizable by sets of ERCs (they are antimatroids, see Riggle (2009), Merchant and Riggle (2014)) while the latter does not.[7] These two definitions need not produce the same factorial typology for a given system, even though here, in BL, the two coincide.

The eleven languages of BL are given in (5). The derivations for each input are given along with a representative total order. In the chart, to save space, the constraint *[+Low] is represented as *[+L], *[-ATR, -Low] is represented as *[-AL], and the constraint Have[-ATR, -Low] is represented as H[-AL].

[7] In fact, it can be shown that with the latter definition, for any partition of n! total orders there is an HS system of n constraints whose factorial typology's grammars correspond to exactly that partition.

(5) Languages of BL

Lg	Input /a/	Input /A/	Input /h/	Input /H/	Representative Total Order
L1	a→ h → H → H	A→ H → H	h→ H → H	H→ H	*[+L]≫*[-AL]≫Id[Low]≫H[-AL]≫Id[ATR]
L2	a→ h → h	A→ H → h → h	h→ h	H→ h → h	H[-AL] ≫*[+L]≫*[-AL]≫Id[Low]≫Id[ATR]
L3	a→ h → h	A→ H → H	h→ h	H→ H	*[+L]≫Id[ATR]≫H[-AL] ≫ *[-AL] ≫Id[Low]
L4	a→ h → h	A→ A	h→ h	H→ h → h	H[-AL]≫*[-AL] ≫Id[Low] ≫Id[ATR]≫*[+L]
L5	a→ h → h	A→ A	h→ h	H→ H	Id[ATR]≫H[-AL] ≫ *[-AL]≫Id[Low]≫*[+L]
L6	a→ a	A→ H → H	h→ H → H	H→ H	*[-AL]≫H[-AL] ≫ *[+L] ≫Id[Low]≫Id[ATR]
L7	a→ a	A→ H → H	h→ a → a	H→ H	*[-AL]≫Id[ATR]≫H[-AL] ≫ *[+L] ≫Id[Low]
L8	a→ a	A→ A	h→ H → H	H→ H	*[-AL]≫Id[Low] ≫H[-AL] ≫ *[+L]≫Id[ATR]
L9	a→ a	A→ A	h→ a → a	H→ H	*[-AL]≫Id[ATR]≫H[-AL] ≫ Id[Low] ≫*[+L]
L10	a→ a	A→ A	h→ h	H→ h → h	Id[Low]≫H[-AL] ≫ *[+L] ≫*[-AL]≫Id[ATR]
L11	a→ a	A→ A	h→ h	H→ H	Id[Low]≫Id[ATR]≫H[-AL] ≫ *[-AL] ≫*[+L]

(6) Outputs of the languages of BL

Lg	Input /a/	Input /A/	Input /h/	Input /H/
L1	H	H	H	H
L2	h	h	h	h
L3	h	H	h	H
L4	h	A	h	h
L5	h	A	h	H
L6	a	H	H	H
L7	a	H	a	H
L8	a	A	H	H
L9	a	A	a	H
L10	a	A	h	h
L11	a	A	h	H

2.1 The Punu-Lite Language

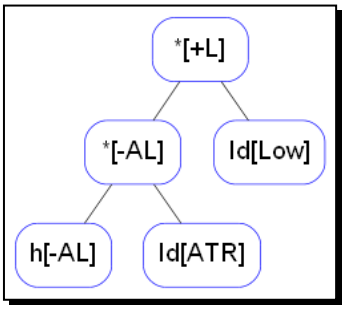
We turn our attention to the first language of the typology, L1, which we rename Punu-Lite. This language shares the property with T&J's Punu that necessary rankings between constraints (both markedness and faithfulness) cannot be determined from the identity map alone. Above in (5) we gave a representative total order of Punu-Lite, repeated in (7) below.

(7) A representative Punu-Lite total order
*[+L] ≫ *[-AL] ≫ Id[Low] ≫ H[-AL] ≫ Id[ATR]

Of interest is not this particular total order for the language, but all total orders that will produce the same derivations that this total order produces. As is often the case, all of a language's derivations can be produced by more than one total order. Learning a

language then can be called successful if the learner selects any of the extensionally and derivationally indistinguishable total orders of the language. Using OTWorkplace, we can identify exactly which total orders produce the same derivations that this total order conjures up. This set is perspicuously represented in the Hasse diagram given below.

(8) Hasse diagram of grammar of Punu-Lite



The total orders consistent with this Hasse diagram give rise to four derivations, one for each input. These derivations are repeated below.

(9) Derivations of Punu-Lite

a. a → h → H → H
b. A → H → H
c. h → H → H
d. H → H

In Punu-Lite, there is only one faithful map H → H, and hence only one locus of information using the Identity Map. Here, as elsewhere, ranking information is determined by comparing a winning candidate to a losing candidate both of which share the same input. For the winning candidate to win, all constraints that prefer the losing candidate must be dominated by some constraint that prefers the winning candidate. This ranking information is encoded in an ERC vector using a three-valued logic: 'W' means a constraint prefers the winner over the loser, '*e*' means a constraint evaluates the two candidates equally, and 'L' means a constraint prefers the loser over the winner.

In PL, the set produced by Gen for the input /H/ consists of three candidates: (H,H), (H,h), and (H,A). Candidates are represented by ordered pairs where the first element of the pair is the input and the second is the output – there is an implied correspondence relation, omitted since the operation set obviates ambiguous correspondence relations. Below are the ERCs produced by selecting the faithful candidate, (H, H), over the two losing candidates.

(10) Information from Identity Map, H → H

Input	winner ~ loser	*[+Low]	*[-AL]	H[-AL]	Id[ATR]	Id[Low]
/H/	H ~ h	*e*	W	L	W	*e*
/H/	H ~ A	W	*e*	*e*	*e*	W

As can be seen, for (H, H) to win over (H, h), either *[-AL] or Id[ATR] must dominate H[-AL]. This fact is encoded in the ERC ⟨*e*WLW*e*⟩, where the order of constraints is identical to those in the comparative tableau in (10). Throughout the paper I retain this order for ERC presentation. It is not meant to imply anything about the learning state, the target language, or any other order information about the constraint. Turning to the fact that H does not map to A, no ranking information is gained since (H, A) is harmonically bounded by (H,H), represented by the ERC ⟨W*eee*W⟩.

The one piece of ranking information gleaned from the identity map, ⟨*e*WLW*e*⟩, grossly underrepresents the totality of ranking requirements that the learner must determine about the target language, given above in (8). In fact, for the learner to successfully learn the target language, correct identification of *all* of maps of the language must occur, none of which, besides H→H, are faithful maps. This is not a typical occurrence in HS systems. Often a single form determines the behavior of many related forms, e.g. the left-to-right assignment of trochaic feet to a four syllable input in many HS systems will determine the parsing of all even-length inputs. Here, due to the simplicity of the system, a more comprehensive inventory of mappings must occur. For example, one of the maps that the learner must discover is that A → H. The learner must discover that that A → H and that A ↛ A from which it can construct the ERC give in (11).

(11) Information from map, A → H

Input	winner ~ loser	*[+Low]	*[-AL]	H[-AL]	Id[ATR]	Id[Low]
/A/	H ~ A	W	*e*	*e*	*e*	L

This crucial bit of information, ⟨W*eee*L⟩, that *[+Low] ≫ Id[Low], is only determinable from this map. The learner needs some principled way of discovering this input and non-faithful map, and extracting this ranking information. The organization of the input space into candidate maps provides the means of finding informative non-faithfully mapped forms.

3. Candidate Maps.

3.0 Distance between forms.

In this section I define the distance between two forms, the local candidate map and (global) candidate map of an HS system. Throughout I assume that the HS systems under consideration have operations that are fully reversible, so that if given two forms f and g

and form f ∈ Gen(g), then g ∈ Gen(f). Nothing crucial relies on this assumption but it streamlines certain expositional areas.

Two forms f and g, f≠g, in an HS system, *H*, are said to have a distance of one between them if and only if g ∈ Gen(f). This means that they are one step apart, where Gen defines what a step is. A form, f, is said to have a distance of zero with itself. What this means is that outputs of the candidates of Gen(f) are one unit away from f, except for f itself, having distance zero from itself. Concretely, in BL, the form H has two forms that are one unit away from it, h and A. In a following section, this notion of distance is shown to be expandable into a metric on all of the forms of the system. I defer that exposition until its utility in the construction of candidate maps is shown.

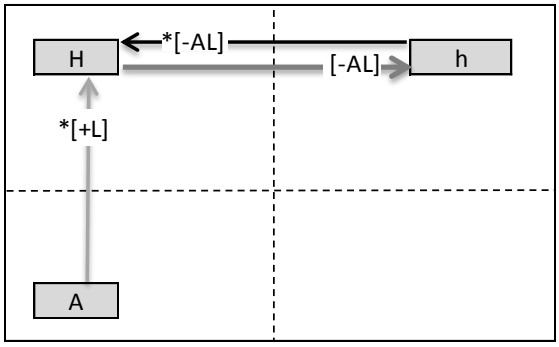
3.1 Local Candidate Maps

Here I first define a *local candidate map*, and then build the candidate map for the system from the set of local candidate maps. For a form, *f*, in an HS system *H*, a local candidate map *loc.map*(*f*) is a graph having labeled nodes, one for each form that has a distance of one from *f* and one for *f* itself. This is equivalent to having one node for each candidate of Gen(*f*), if all operations are reversible and the faithful candidate is always included in Gen(*f*)[8]. Each node, *g*, is then connected to the node *f* by directed, labeled edges, one for each markedness constraint that prefers *g* over *f* or vice versa. The direction of the arrow is determined by constraint preference. If the constraint prefers *f* over *g* then the direction is towards *f*, otherwise it is towards *g*. Nodes are connected by an unlabeled edge to *f* if there are no directed edges already connecting them. Sensu stricto, these are multigraphs[9], though I will continue to use the slightly imprecise term graph throughout.

I omit in the representation the edge connecting the faithful output to itself for perspicuity reasons.

Below is the local candidate map for the BL system for input /H/.

(12) Local candidate map for H.



[8] It is straightforward to modify the definition of local candidate map to include systems in which either the faithful candidate is omitted or operations are not reversible. All of the results presented here manifest also in those systems.

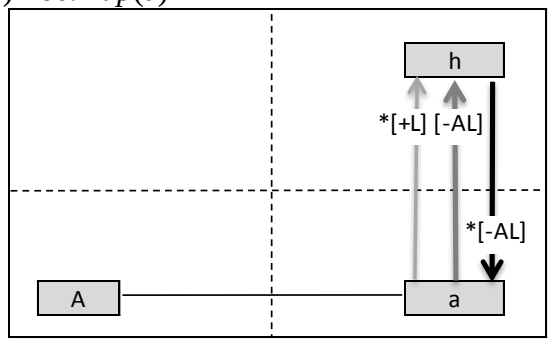
[9] http://en.wikipedia.org/wiki/Multigraph

The graph *loc.map*(H) has three nodes, two for the forms that are one unit away from H, h and A, and one for H. Again, this is equivalent to having one for each of the three candidates produced by Gen(H). Node 'A' is connected to node 'H' by a single directed edge, here rendered in light grey. This arrow is labeled with *[+L] since this constraint assigns one violation to A and zero violations to H. There are two arrows connecting the nodes 'H' and 'h'. The right-to-left arrow labeled *[-AL], rendered in black, corresponds to the constraint *[-AL] preferring the output H over the output h. The left-to-right arrow labeled H[-AL] encodes the fact that H[-AL] prefers h to H.

The dotted horizontal and vertical lines intersecting the arrows are not part of the candidate map, but represent which faithfulness constraints must be violated to map H to its connected brethren. So, for example, to map H to h, the winning candidate, (H, h) must incur a violation of Id[ATR], here encoded in the vertical dotted line. To map H to A, a violation of Id[Low] is accrued to the unfaithful candidate.

Another local candidate map is given for the input a in (13).

(13) *Loc.map*(a)



Here there are three nodes again, one each for the candidates produced by Gen(a). There are three arrows between nodes 'h' and 'a': two of the markedness constraints prefer the output h over a, *[+L] and H[-AL], while *[-AL] prefers a over h. Nodes A and a are connected by a single non-labeled, undirected edge because there are no markedness constraints that prefer either output over the other in the system BL. Faithfulness constraints are represented with dotted lines using the same encoding as in the previous example, crossing a horizontal line represents a violation of Low, and a vertical ATR.

Local candidate maps, and their slightly more articulated cousins, candidate maps, can provide immediate information about individual languages and the typology as a whole. As noted above, the *loc.map*(a) connects nodes a and A only by an unlabeled edge. This means that no constraint prefers the unfaithful mapping a → A over the faithful mapping a → a. An analyst (and a learner) can conclude that no language in the typology will map a → A or vice versa.

Furthermore, knowing a mapping along with the candidate map can yield ranking information. Suppose that a → h. This means that the candidate α = (a, h) must beat β =

(a, a). The local candidate map represents, via directional arrows, which constraints prefer α to β and vice versa – we can see immediately from (13) above, that H[-AL] and *[+L] prefer α over β while *[-AL] and Id[Low] prefer β over α. This information is immediately encodable in ERC form: record W's in those constraints that prefer the winner to the loser according to the map, L's in those preferring the loser to the winner, and *e*'s for constraints absent from the comparative link. Here that translates to the ERC ⟨WLW*e*L⟩.

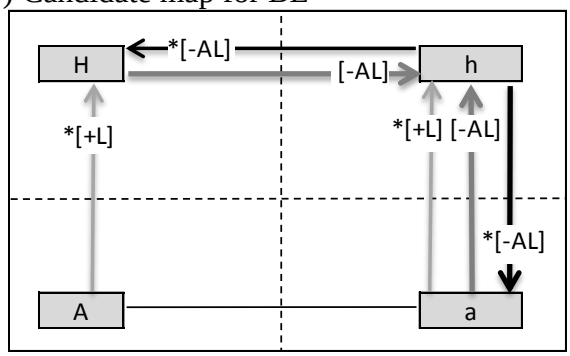
Besides presenting a perspicuous representation of available candidates the local candidate map can serve as useful learning tool. The local candidate map is constructible by the learner. Given a single input, the local candidate map is produced by Gen and Con, with no need to look further amongst the multitudinous search space. Also, a local candidate map for an input is language neutral: any learner of any language will produce the same candidate map for a given input.

Intriguingly, local candidate maps can be constructed in a wide variety of Optimality Theoretic systems. A candidate map for an input relies only on having the outputs of Gen and the markedness constraints' relative evaluation of the input and the output forms. For OT systems in which the input can serve as an output, candidate maps can be built. Since the non-harmonically bounded set of candidates for an input is always finite in OT (up to violation profile identity), candidate maps in OT will be (relatively) tractable finite graphs. Being able to build local candidate maps in OT also means that all other tools developed here are germane to OT systems. Global candidates maps and recursive join learning sit on the foundation of local candidate maps, hence are importable to OT.

3.2 (Global) Candidate Maps

A candidate map for a typology is the graph union of all of the local candidate maps for all inputs of the typology. The graph union is simply the union of all the nodes and edges of each of the local candidate maps of the system. For BL, the candidate map consists of a single connected graph, having four vertices, one for each input to the system. In systems without epenthesis and deletion and inputs consisting of segments of different lengths the candidate map may not be connected, as there may be no path of single steps between forms of different length. For systems with epenthesis and deletion they may not be finite.

(14) Candidate map for BL



Note that it is the graph union of only the two previously shown local candidate maps. Adding in the local candidate maps for h and A adds no new edges nor nodes since they already are extant in the other local candidate maps.

Thus the candidate map in (14) is the candidate map for the full BL system – candidate maps do not vary by language and are fixed, once the system is defined. From this map we can start to deduce properties of the typology. As noted above A and a are only connected by an undirected edge which only obtains when there are no constraints that prefer one output form over the other. This means that in no language in the typology will A map to a or vice versa. Following similar lines, no language will map H to A as there is no directed edge pointing in that direction.

Global candidate maps can be much harder to visualize than the one presented here. For example, in a system with two segment inputs, each segment consisting of binary features and the operations being flipping of single binary features, the candidate map will have a four-dimensional hyper-cube structure. (This is the structure of the candidate map for T&J Punu system, if one conflates non-initial segments into one segment.) Even so, portions of the global candidate map are often easy to manipulate, and local candidate maps are guaranteed to be planar graphs, falling well-within the ocular resolution of our primate visual system, and such ocularly resolvable systems can be algebraic manipulated in a straightforward manner.

3.2 A metric space of forms

Arranging the forms of an HS system into a candidate map graph suggest that there is an inherent notion of distance between two forms. Indulging in a small number of formal definitions, I show that an HS system imposes a metric structure, a well-defined notion of distance, on all the forms in the system.

A given HS system, *H*, has a set of forms, F, that can serve as inputs and outputs. This is the set upon which the metric will be defined.

Definition. Given a set of candidates, C, define *out*(C) to be the set of output forms of the candidates of C. Also, for a single candidate, cand, *out*(cand) is the output form of the candidate.

So for a set of candidates, C, *out*(C) = $\bigcup$ *out*(cand), where cand$\in$C.

Definition. Given a set of candidates, C, define *in*(C) to be the set of input forms of the candidates of C. Also, for a single candidate, cand, *in*(cand) is the input form of the candidate.

Both functions *out* and *in* are overloaded, in the computer science lingo, to represent two functions, one whose domain is the power set of candidate, the other domain is the set of candidates.

Definition. Given candidate $c \in$ C the set of candidates, where *in*(*c*) = f, define ERCs(*c*) = {$c \sim d \mid d \in$ Gen(f)}.

For a candidate *c*, ERCs(*c*) are those ERCs produced by making *c* the optimal candidate in its candidate set.

Definition. A *path* between f, g $\in$ F, p(f,g), is a finite sequence of forms, $f_0, f_1, \ldots, f_n$, such that f_0 = f and f_n = g, and $(f_i, f_{i+1}) \in$ Gen(f_i) for i = 0, 1, 2, …, n-1.

Note that this definition of a path does not require that it be realized as a derivation in any language in the typology. For example, in BL, the two element sequence, (A, a), is a path starting at A and ending at a. But there is no derivation in which A maps to a in any language in the system. Even so, this definition will be useful in defining distance.

Throughout we assume that the HS systems under consideration have operations that are fully reversible, so that if given two forms f and g and form f $\in$ Gen(g), then g $\in$ Gen(f).

This ensures that if p(f,g) is a path between f and g, then there is a path p(g,f) starting at g and ending at f. This assumption simplifies many aspects of the discussion and comports with all of the HS systems discussed, but is by no means needed for the results to stand with minor modifications.

Definition. Given a path p(f,g) = $p_0, \ldots, p_n$, the *length* of p(f,g), |p(f,g)| = n.

We now have the pieces to define a metric on the forms of any HS system.

Definition. For f, g $\in$ F, define d:F $\times$ F $\rightarrow \mathbb{R}$ by

- d(f,g) = min{|p| : p is a path between f and g}, if a path between f & g exists.
- d(f,g) = max{|p| : p is a path between any j,k $\in$ F} + 1, if no path between f & g exists.

First, note that this definition aligns with the definition given above in constructing a candidate map. We now show that this defines a metric on the set of forms, by showing that d(x,y) has the four defining properties of a metric.

Claim. This is a metric on F.
Proof. We first need to prove that d(f,g) is non-negative for all f, g $\in$ F. This is obvious from the definition, as paths cannot have a negative length.
(Identity of indiscernables). Given two forms f, g $\in$ F, d(f,f) = 0 since the one element sequence (f) is a path between between f and itself. Now suppose d(f,g) = 0. Then there is a path (f_0) between f and g. But by the definition f = f_0 = g.
(Symmetry) If p(f,g) = a_0, …, a_n is a path between f and g, then a_n, …, a_0 is a path between g and f.
(Triangle Inequality: d(f,g)$\leq$ d(f,h)+d(h,g)) Let f, g, h $\in$ F. Let p_1 be a minimal path between f and h and let p_2 be a minimal path between h and g, if such paths exist. There are two cases. (1) The two paths exists. When concatenated, p_1p_2, is a path between f and g. It may be minimal or it may not. Either way the triangle inequality holds. (2) There is no path between either f and h or between h and g. Without loss of generality, assume that no path between f and h exists. Then there may or may not be a path between f and g. Regardless, d(f,g) $\leq$ d(f,h) + d(h,g) since d(f,h) is the maximal path length plus one. QED.

So, there is a metric space associated with each Harmonic Serialism system. This is a metric space on the forms of the languages in the system. This is equivalent to the geodesic distance on the candidate map graph of the forms of the system. The only difference is that the geodesic distance uses infinity as the distance between nodes that are not connected by a path. The geodesic distance between nodes on a graph is the shortest path between those nodes – since the form metric ignores derivational viability of paths the two notions coincide.

This metric space on the forms contrasts with the metric space induced on the *grammars* of the languages in an HS system using the Riggle Metric (Riggle 2012, see Merchant & Prince 2014 for this idea developed in OT.)[10]

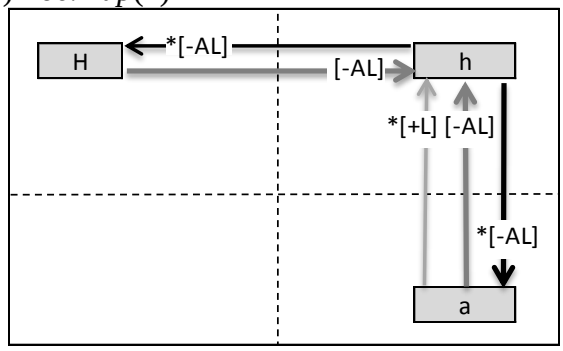
4 Learning using the Join

[10] The Riggle Metric and its induced topological space on the languages are once again different from the *Phenotype Space* present among languages in a typology. Grammars (sets of total orders) are directly analogizable as RNA sequences, and languages (sets of derivations) as the phenotypic expressions of a set of total orders. It is well known that a single change to an RNA sequence, say cytosine mutating to guanine, often causes no change in the phenotype. This is similar to flipping the order of two non-crucially ranked adjacent constraints and producing the same set of derivations. Biologists use the fact that a large set of RNA sequences produce the same phenotype to define a topology on the phenotypes of a system. Roughly, phenotype α is close to phenotype β if making a single random mutation in an arbitrary RNA sequence that produces α has a high probability to now produce β. (see Stadler et al. (2001) and Stadler and Stadler (2004) for discussion) This notion of distance is immediately importable to HS and OT languages by asking what is the probability of moving from language α to language β by swapping a pair of adjacent constraints in a randomly selected total order that produces α. Assuredly, an Erlangenian-type program could further our understanding of the interrelations of the topological structures we face.

Often, knowing that a given form does not map to itself can yield informative ranking information about the target language even if what the form maps to is not known. In this section I show that ranking information from non-faithful maps, useful for learning, can be extracted using the join (Merchant 2008).

Consider the situation faced by the learner attempting to learn Punu-Lite. In this language the input /h/ does not map to itself. Constructing the local candidate map for /h/ yields (15).

(15) *Loc.map*(h)



Reading off the local map for h: for the map h → H to obtain both Id[ATR] and H[-AL] must be dominated by *[-AL]. This is representable by the ERC ⟨*e*WLL*e*⟩. Looking to the other possible map, h → a, this map occurs when *[-AL] dominates the three constraints H[-AL], *[+L], and Id[Low], which is ERC-encodable by ⟨LWL*e*L⟩. Notice that for either of these mappings to obtain, *[-AL] must dominate H[-AL]; that is ⟨*e*WL*ee*⟩. This is the useful ranking information that the learner can determine by knowing that h does not map to itself.

The ranking requirements for a non-faithful mapping in this situation are exceptionally clear because of the symmetric construction of the markedness constraints: *[-AL] prefers forms without an h while H[-AL] has an opposite set of preferences. In a typical configuration of a non-faithful mapping, the interplay between the possible set of mappings and the rankings that produce them may not be so perspicuously arranged.

The question of determining ranking information from a non-faithful map can be reformulated in terms of ERCs. Here: what can we say about the target language knowing only that either ⟨*e*WLL*e*⟩ or ⟨LWL*e*L⟩ is true? The answer is also in terms of ERCs. We know that ⟨*e*WL*ee*⟩ must be satisfied in the target language. This latter ERC contains the only L shared between the two basic ERCs.

Broadening this Q&A to any set of ERCs we ask: given a collection of sets of ERCs, each of which delimits a set of total orders (at least one of which is true of the target language, but not necessarily all of them), is there a set of ERCs that delimits the union of the total order of the sets of ERCs, does so minimally, and if so, what is it? Minimality here means that no other set of ERCs defines a proper subset of total orders while still

encompassing the union. This question has a definitive answer: the join provides exactly such an ERC set.

As shown in Merchant 2008, the join of two ERCs is a single ERC that delimits the smallest set of ERC-representable total orders that includes the union of the total orders of the two joinard ERCs. The join of two ERCs, α and β, denoted α+β, is similar to logical disjunction: the disjunction of two logical statements is true when either of the statements are true. The join of two ERC statements is satisfied by any total order that satisfies either of the two constituent ERCs. The join is the smallest such set determinable by an ERC – in some situations the straight union of total orders of two ERCs is not ERC-representable.

The join of two ERCs, α and β, is produced by component-wise joining of the {W, e, L} values in the respective ERCs. This is done by placing the total order W > *e* > L on {W, e, L} and then joining elements from this set using the lattice-theoretic join. This is similar to logical disjunction with the order T > F, where W corresponds to T and L to F. Joining shares with logical disjunction idempotency (X ∨ X = X) and symmetry (X ∨ Y = Y ∨ X). This is spelled out further in (16).

(16) Join over set {W, e, L}
W ∨ X = W
e ∨ L = *e*
L ∨ L = L

Applied to the example above we can algorithmically produce the ERC-representable ranking requirements determined by the disjunction of the two ERCs.

(17) Join of ERC_1 = ⟨*e*WLL*e*⟩ and ERC_2 = ⟨LWL*e*L⟩

	*[+Low]	*[-AL]	H[-AL]	Id[ATR]	Id[Low]
ERC_1	***e***	**W**	**L**	**L**	***e***
ERC_2	**L**	**W**	**L**	***e***	**L**
ERC_1+ERC_2	***e***	**W**	**L**	***e***	***e***

Merchant (2008) provides a means of joining together sets of ERC sets, not just pairs of ERCs – this is not needed in this example because of the structure of the candidate sets under consideration. Each possible non-faithful map produces only one informative ERC. The only informative losing candidate is the faithful candidate; if the learner creates an ERC from two non-faithful candidates, say α and β, a winner must be chosen, creating an ERC, say α~β. But the learner does not know that α beats β and must therefore also create the ERC β~α. The join of these two ERCs places no restrictions on the total orders of the language as their disjunction includes all total orders.

If the learner can rule out one or more of the non-faithfully mapped candidates then the other non-faithful forms may provide more than one informative ERC. In this situation

the set-join of ERCs can be used to gather the rankings that produce the non-ruled out, non-faithful maps.

5. Using the map and the join for learning.

The algorithm proposed here, RJL, uses both local candidate maps and the join to determine ranking information not present in the fully-faithful maps of a target language. RJL starts by using the identity map to learn ranking requirements (Prince & Tesar 2004) producing a set of ERCs that must be satisfied in the target language. This set of ERCs becomes the data store, a data store that is added to with further ERCs produced in the course of the algorithm.

Starting with the observable (and hence fixed) forms of the language, for each of these fixed forms, the RJL constructs local candidate maps for all non-fixed forms one unit away from the fixed forms. Each non-fixed map is checked for consistency with the data store. Those maps that are consistent with the data store, that is, those which the language could produce, are joined together – extracting shared ranking information, ranking requirements that is then checked against the data store for consistency. The join, represented as an ERC, is added to the data store. This ERC is marked as being produced from a possible non-faithful map.

In this presentation we assume that the learner is presented with all forms that are faithfully mapped, to focus on the RJL and the candidate maps it uses. Relaxing this assumption requires some note about how inconsistency and indeterminacy are dealt with, a number of possibilities obtain. One such: if latter steps are inconsistent with the data store and this ERC is identified as a locus of inconsistency, using say, RCD, it could simply be excised from the data store.

The algorithm then repeats, producing local candidate maps for forms that are one unit away from those just processed, excluding fixed forms or forms previously processed. The algorithm terminates when a predetermined distance has been reached or when all forms have been processed.

In more formal prose, the algorithm is given in (18).

(18) Recursive Join Learning algorithm

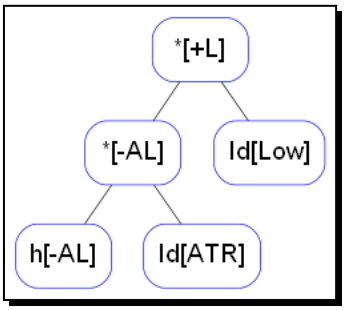
- Let FF = set of output forms learner is exposed to, the fixed forms
- Set DS = $\emptyset$, the data store of ERCs consistent with the target language
- **Step 1. Learning from the identity map**
    - For each f $\in$ FF,
        - Set DS = DS $\cup$ ERCs($cand_f$), where $cand_f$ = (f,f)
- Set CF = FF, the current forms, and PF = FF the processed forms.
- Let k = 1, the distance from the FF currently being processed.
- **Step 2. RJL**
    - For each f $\in$ CF,
        - Let $D_k(f) = \{g \in F \mid d(f,g) = k \text{ and } g \notin PF\}$

- For each $g \in D_k(f)$
  - Construct *loc.map*(g)
  - Let G= Gen(g) – (g,g), J.ERCs = ∅
  - For each $\alpha \in G$, if α~(g,g) is consistent with DS
    - Add α~(g,g) to J.ERCs
  - If DS entails join(J.ERCs), remove g from $D_k(f)$
  - Set DS = DS ∪ join(J.ERCs)
- Remove f from CF if $D_k(f) = \emptyset$.
- Set PF = PF ∪ $D_k(f)$
- Increment k by one and repeat step two if CF ≠ ∅.

5.1 Applying the algorithm to Punu Lite.

Here we apply the algorithm to Punu Lite. The ranking requirements for Punu Lite were given in (8), repeated here in (19).

(19) Punu-Lite ranking requirements



The algorithm starts with Identity Map Learning. PL has only one fixed form, H → H, and so only one set of ERCs are produced using the Identity Map. As discussed above, there is only one informative ERC produced from the identity map, shown below.

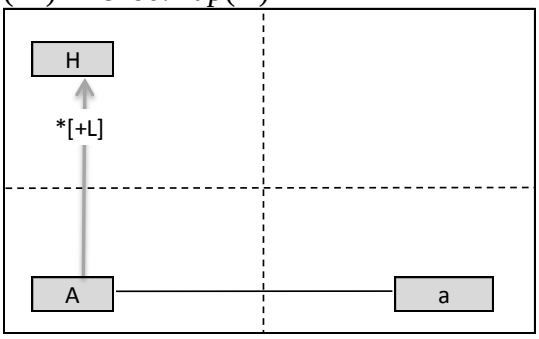
(20) ERC produced by Identity Map Learning

	*[+Low]	*[-AL]	H[-AL]	Id[ATR]	Id[Low]
(H,H)~(h,H)	*e*	W	L	W	*e*

From the sole fixed map of the language one ERC is determined ⟨*e*WLW*e*⟩. This is added to the data store, DS = {⟨*e*WLW*e*⟩}. The learning algorithm then proceeds to the next stage of learning – constructing the local candidate maps for forms that are one unit away from the form H.

There are two forms one unit away from the fixed point H; these are h and A. We first turn our attention to A. The algorithm constructs *loc.map*(A), given below.

(21) The *loc.map*(A)



There are two forms that are one unit away from A, and so the *loc.map*(A) consists of these two forms plus the faithful form A. We can see from the *loc.map*(A) that the map A → a will provide no ranking information for the learner since this map is harmonically bounded by the faithful candidate. No constraint prefers the output a over A, while the faithfulness constraint Id(ATR) prefers A over a.

The map A → H is informative. There is one markedness constraint *[+Low] that assigns no violations to H and one to A. There is one faithfulness constraint, Id[Low] preferring the faithful candidate. From this the learner gets the ERC in (22), ⟨W*eee*L⟩. Note that at this stage, there is only one ERC produced from *loc.map*(A) that is consistent with the data store, since the map A→a is inconsistent with *every* ranking. Because there is only one possible mapping, there is no joining of the ERCs produced from the local map.

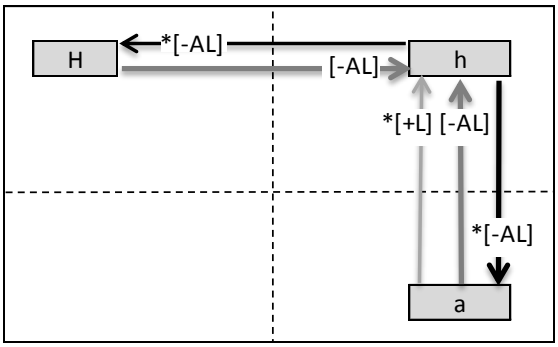
The data store consists now is DS = {⟨*e*WLW*e*⟩, ⟨W*eee*L⟩}.

(22) ERC from A → H map.

/A/	*[+Low]	*[-AL]	H[-AL]	Id[ATR]	Id[Low]
H	0	0	0	0	1
H~A	W	*e*	*e*	*e*	L

Now, moving to the second form one unit away from the fixed form, we get *loc.map*(h).

(23) The *loc.map*(h)



As we have seen above, there are two non-faithful maps here, the learner does not know which one is true of the target language. Even though there is ambiguity here, there is ranking information determinable, in ERC form, from these two maps.

(24) ERCs and join of ERCs from non-faithful /h/ maps.

	*[+Low]	*[-AL]	H[-AL]	Id[ATR]	Id[Low]
ERC_1=(H,h)~(h,h)	***e***	**W**	**L**	**L**	***e***
ERC_2=(A,h)~h,h)	**L**	**W**	**L**	***e***	**L**
ERC_1+ERC_2	***e***	**W**	**L**	***e***	***e***

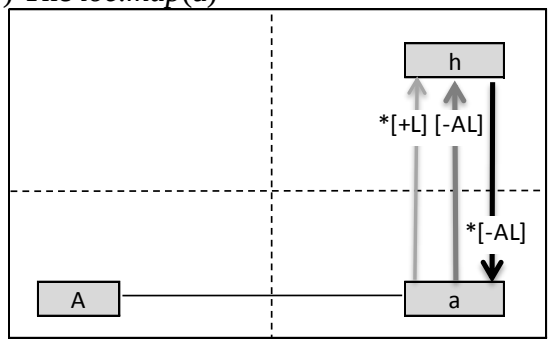
The join of the two ERCs produces the ERC ⟨*e*WL*ee*⟩. The fact that *[-AL] must dominate H[-AL] can only come from this mapping – no other mappings provide this information. This ERC is added to the data store.

Note that this ERC entails one of the ERCs already present in the DS. The ERC gotten from the faithful map, H→H, ⟨*e*WLW*e*⟩, says that either *[-AL] or Id[ATR] must dominate H[-AL]. The join-produced ERC removes the disjunction: the language must have *[-AL] dominating H[-AL]. I omit the less informative ERC from the DS (no rankings were harmed in this omission). The data store now consists of DS = {⟨*e*WL*ee*⟩, ⟨W*eee*L⟩}.

After gathering the ranking information from *loc.map*(A) and *loc.map*(h), the algorithm proceeds to determine the forms that are one unit away from A and h (and therefore two units away from H). The one form that has not been analyzed that is one unit away from A is the same form that is one unit away from h, namely a. The algorithm is blind to the fact that the form a can be reached via two paths, either through h or A. Regardless, form a is next on the docket.

Below is *loc.map*(a), repeated from above.

(25) The *loc.map*(a)



As mentioned above, the map a → A can easily be seen to be uninformative since no markedness constraint prefers A over a. The map a → h differs. A single ERC, true of the target language, is extracted by the learner. This is given below.

(26) ERC from a → h beating faithful map.

	*[+Low]	*[-AL]	H[-AL]	Id[ATR]	Id[Low]
(a,h)~(a,a)	W	L	W	*e*	L

This ERC, ⟨WLW*e*L⟩, is true of the target grammar, as is ⟨*e*WL*ee*⟩ gotten from the join of the non-faithful maps from h. The second ERC requires *[-AL] above H[-AL] ensuring that the first ERC can only be satisfied by rankings in which *[+Low] dominates the L's in the ERC. This is easily seen using the fusion of the two ERCs, shown in (27). We augment the DS with this ERC, entailed by the two. (See Prince 2002 and Brasoveanu & Prince 2011 for discussion of fusion). DS = {⟨*e*WL*ee*⟩, ⟨W*eee*L⟩, ⟨WLL*e*L⟩}.

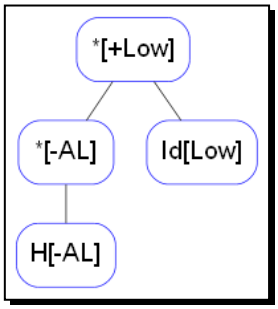
(27) Fusion of two ERCs

	*[+Low]	*[-AL]	H[-AL]	Id[ATR]	Id[Low]
ERC_1	*e*	W	L	*e*	*e*
ERC_2	W	L	W	*e*	L
$ERC_1 o ERC_2$	W	L	L	*e*	L

The algorithm is now finished: there are no further forms to be considered.

We can now ask how well did the algorithm perform in learning. The algorithm succeeded in finding some of the ranking information determinable only from non-faithful maps. The ranking information gleaned is presented in (28). This is the Hasse representation of the ERCs in the data store, DS = {⟨*e*WL*ee*⟩, ⟨W*eee*L⟩, ⟨WLL*e*L⟩}.

(28) Hasse diagram of ranking information produced from algorithm



The sharp-eyed reader will note that Id[ATR] has been omitted from the diagram. This is because the algorithm did not determine any ranking information about Id[ATR]. Recall that in the target language Id[ATR] must be ranked below *[-AL].

The reason this information was not determined, even though the correct grammar ranks Id[ATR] below *[-AL], is that this crucial ranking is only determinable by knowing which of a set of non-faithful maps obtains in the target language. The algorithm only extracts information by disjunctively comparing non-faithful maps to faithful maps. Knowledge of which non-faithful map occur in the target language would provide the learner with this crucial bit of information. Here the ranking information is revealed by selecting the non-faithful h→H over the faithful h→h, shown in (29). This information is obscured because when the learner considers this map, the fact that h does not map to a cannot be excluded.

(29) Missing ranking information from h→H

	*[+Low]	*[-AL]	H[-AL]	Id[ATR]	Id[Low]
(h,H)~(h,h)	*e*	W	L	L	*e*

This last piece of information can only be extracted if the map h→a can be ruled out. Indeed it can be. In the final step of the algorithm, the *loc.map*(a) is constructed and ranking information is determined from its non-faithful maps. In particular, the learner determines that a must map to h, since a is non-faithfully mapped and no language can map a→A. Any language in which a→h cannot also have the map h→a, as this would induce a cycle, preventing convergence of derivations that included both maps.

This points out that the intuition one might have that unexplored forms nearest the explored forms will likely have more information for the learner than more distal forms (call this the Proximity Principle) may not always be true. In this case, the non-faithful form a can map only to the form h, and as such, other non-faithful more proximate maps to the starting point must follow.

6. Conclusion

Facing the HS learner are two interrelated problems: how to select non-faithful forms to determine ranking information from, and how to determine ranking information from possibly non-faithful forms. In this paper I have shown that Harmonic Serialism systems inherently organize the forms of the typology with a distance metric where forms are close to one another if there is a short sequence of candidates starting at one and leading to the other. This distance metric can be used to organize the forms of the space into local candidate maps – graphs that represent both which forms a given form could map to, and what the ranking requirements are for a given map to obtain. These local maps provide a powerful learning tool, answering the first question. The forms that one should investigate first are those that are closest to the fixed forms. These are likely to be informative since they are likely to not map to themselves and they are likely to map to the fixed forms, being close.

The local maps are then utilized by a second idea presented here. The join of ERCs can provide useful ranking information even in the face of ambiguity. A learner does not know which of a set of mappings may obtain. But the learner does know that at least one of the set must be correct. The join exploits this fact by producing an ERC that includes the rankings of each of the ERCs joined together. Because this holds, the learner can be confident that the ERC produced by the join holds for the target language.

These two ideas are then put together in the Recursive Join Learning algorithm. This algorithm begins by extracting ranking information from faithfully mapped forms. It then finds those forms that are one unit away from the faithful forms. Local candidate maps are constructed for each, and the join is used to determine ranking information shared by each of the non-faithful maps. The algorithm then recurses on the forms one unit away from those considered.

This algorithm exploits the inherent structure present in every HS system by organizing forms by distance from the fixed forms of the language. It is a surprising result that there is coherent distance metric on the forms of an HS system. Even more surprising is that this notion of distance (its concomitant tools) is immediately exportable to many instantiations of OT. There appears to be a rich set of structures on both OT and HS languages, currently minimally explored. Likely other learning questions will be answered by expanding our understanding of the relationships amongst them.

References.

Apoussidou, Diana (2007). The learnability of metrical phonology. PhD dissertation, University of Amsterdam.

Bennett, William (2013). Dissimilation, Consonant Harmony, and Surface Correspondence. PhD dissertation. Rutgers University.

Brasoveanu, Adrian and Alan Prince (2011). Ranking and necessity: the Fusional Reduction Algorithm. Natural Language and Linguistic Theory 29:3-70.

Hayes, Bruce (2004). "Phonological acquisition in Optimality Theory: the early stages". Appeared 2004 in Kager, Rene, Pater, Joe, and Zonneveld, Wim, (eds.), *Fixing Priorities: Constraints in Phonological Acquisition.* Cambridge University Press.

Hyman, Larry (2002). Is there a right-to-left bias in vowel harmony? Paper presented at the 9th International Phonology Meeting, Vienna. Available (January 2014) at http://linguistics.berkeley.edu/~hyman/Hyman_Vienna_VH_paper_forma.pdf.

Hyman, Larry (2008). Enlarging the scope of phonologization. UC Berkeley Phonology Lab Annual Report. 382–408.

Jarosz, Gaja (2006). Rich lexicons and restrictive grammars: maximum likelihood learning in Optimality Theory. PhD dissertation, Johns Hopkins University.

Kager, René, Joe Pater & Wim Zonneveld (eds.) (2004). *Constraints in phonological acquisition.* Cambridge: Cambridge University Press.

Kwenzi Mikala, Jerome (1980). Esquisse phonologique du punu. In F. Nsuka-Nkutsi (ed.) E´léments de description du punu. Lyon: CRLS, Université Lyon II. 7–18.

McCarthy, John (2000). Harmonic serialism and parallelism. *Proceedings of the North East Linguistics Society.*

Merchant, Nazarré (2008). Discovering Underlying Forms: Contrast Pairs and Learning. PhD dissertation. Rutgers University. ROA-964.

Merchant, Nazarré (2011). Learning ranking information from unspecified overt forms using the join. *Proceedings of the Chicago Linguistic Society*. ROA-1146

Merchant, Nazarré (2014). Effects of Candidate Omission in Harmonic Serialism. Ms. Eckerd College.

Merchant, Nazarré and Alan Prince (2014). The Mother of All Tableaux. Ms. Eckerd College & Rutgers University.

Merchant, Nazarré and Jason Riggle (2014). OT Grammars, Beyond Partial Orders: ERC Sets and Antimatroids. *Natural Language and Linguistic Theory* (to appear).

Prince, Alan (2002). Entailed Ranking Arguments. Rutgers University. ROA-500.

Prince, Alan & Paul Smolensky (1993). Optimality Theory: constraint interaction in generative grammar. Ms, Rutgers University & University of Colorado, Boulder. Published 2004, Malden, Mass. & Oxford: Blackwell.

Prince, Alan & Bruce Tesar (2004). Learning phonotactic distributions. In Kager et al. (2004). 245–291.

Prince, Alan, Bruce Tesar & Nazarré Merchant (2014). OTWorkplace. Available at http://ruccs.rutgers.edu/~prince/.

Riggle, Ranking and Convexity (2012). Talk at Rutgers Optimality Research Group.

Riggle, Jason (2009). "The Complexity of Ranking Hypotheses in Optimality Theory". Computational Linguistics 35(1): 47–59

Smolensky, Paul (1996). On the comprehension/production dilemma in child language. *Linguistic Inquiry* 27. 720–731.

Stadler, BMR and Stadler PF (2004). "The Topology of Evolutionary Biology" in Ciobanu, G and Rozenberg., eds., *Modeling in Molecular Biology*, Natural Computing Series, New York: Spring-Verlag, 267-286.

Stadler, BMR, Stadler PF, Wagner, G. and Fontana, W (2001). "The topology of the possible: Formal spaces underlying patterns of evolutionary change" *Journal of Theoretical Biology 213*, 241-247.

Tessier, Anne-Michelle (2013). Error-driven learning and Harmonic Serialism. NELS 42:2. 545–558.

Tessier, Anne-Michelle and Karen Jesney (2014). Learning in Harmonic Serialism and the necessity of a richer base. Phonology 31:155–178.