CHAPTER IV

THE GRADUAL LEARNING ALGORITHM ALTERNATIVE

1. An introduction to the Gradual Learning Algorithm

Part of the argument put forward in the previous chapters is that an OT learner can learn partially from the set of available data, and in a frequency-sensitive way, while still using a classic OT grammar that does not encode frequency itself. In this chapter, I discuss an alternative approach in the OT literature that takes a very different view. This method is called the Gradual Learning Algorithm or GLA (Boersma, 1997; Boersma and Hayes, 2001; Curtin and Zuraw, 2001; Levelt and van der Vijver, 2004; Hayes and Londe, 2006). The GLA is fundamentally different than BCD – both in the kind of grammar that it learns, and the way it processes errors – but it has properties that make it very relevant to the issues discussed here so far.

In this section I will introduce the kind of grammar that the GLA learns – a numerical and stochastic brand of OT – and then the GLA algorithm itself. I will highlight how the GLA is inherently a stage-like learner, and consider the role of ranking biases in its workings. Note that this section is not intended to provide a comprehensive introduction to the GLA: see Boersma (1998); Boersma and Hayes (2001).

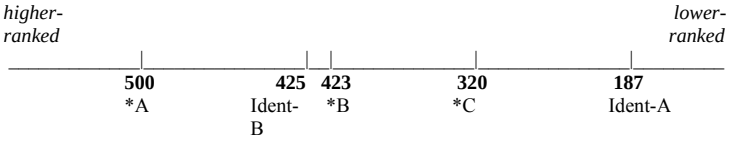
1.1 The GLA view of constraint rankings

The constraint rankings that the Error-Selective BCD learner learns are classic OT hierarchies in that its constraint rankings are *ordinal.* Two constraints in the true, classic OT of Prince and Smolensky (1993) can only stand in one of two relations – A >> B or B >> A – and there is no sense in which A can be ranked *more or less* above B or vice

versa. We have also seen that in the T/S view of learning, this property is relaxed slightly to allow a third relation: one of equal ranking. Thus, the first constraint ranking learned by the BCD algorithm is of the form {M} >> {F}. In this hierarchy: for each M and F constraint pair, M >> F, and within the M and F strata, each constraint is ranked equally with all others.

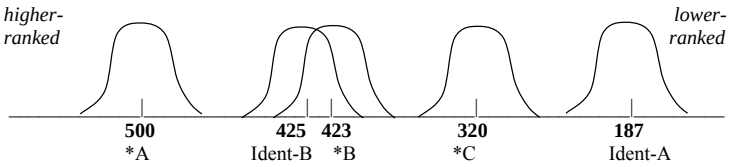
In contrast, the constraint rankings that the GLA learns are what I will call *numerical* rather than ordinal. The GLA learns grammars where constraints have ranking values along a number line, so that every constraint is ranked not just above or below every other, but *at a certain distance* above or below every other. This is shown below for some hypothetical constraints and ranking values:

1) *The numerical view of OT constraint ranking (first try)*



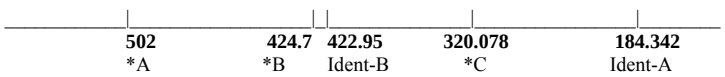
Furthermore, the GLA learner also assumes that constraint rankings are *stochastic* – that they are perturbed by some statistical noise. This noise is introduced by assuming that a constraint's ranking value does not just represent its single point on the scale, as in 1), but rather the midpoint of a normal (Gaussian) distribution of values. This menas that constraint X's ranking value is the place in the hierarchy that X is the most likely to sit, and the further away from X's ranking value you get, the less likely X is to have that value at any point in time:

2) *The stochastic view of OT constraint rankings*[1]



Each time a stochastic OT grammar is used, a single value is chosen for each constraint from its distribution of values; this choice creates a scale of single-point constraint values as in 3) below, which for practical purposes can be used by EVAL as a classic OT ranking:

3) *A one-time ranking*



4) *The ordinal version of 3)*
*A >> * B >> Ident-B >> *C >> Ident-A

(For reasons of typographic ease the GLA numerical rankings I draw from now on will not contain the normal distribution curves above each ranking value, but the GLA model I will be discussing throughout does indeed include this stochastic component – crucially so when treating variation in section 6.)

Despite the fact that each run of this grammar relies on a single ranking that can be equated with a classic OT hierarchy as in 4), there are crucial differences between the ordinal and numerical OT versions of OT. The example above has already demonstrated

[1] Note that the curves I have been able to draw freehand here are really not in the shape of a normal distribution at all.

this, with respect to the ranking of *B and Ident-B. The ranking values in 1) showed us that in this grammar Ident-B is ranked above *B, but only slightly above; this means that in 2), their distribution of values overlap considerably. Because of the stochastic component of this model, these similar ranking values mean that Ident-B is only slightly more likely to outrank *B in any run of the grammar: in the one-time ranking in 3), for example, the value chosen from *B's distribution is in fact *higher* than the one chosen from Ident-B's, so that for this use of the grammar, their ranking has been reversed. (Note that the amount to which the curves of two constraints overlap is a function not only of how similar their ranking values are but also how much random noise the system uses to choose one-time values.)

It is in this way that the relative distance between constraints makes numerical, stochastic OT different from the classic theory of Prince and Smolensky (1993/2004). It is also the conception of ranking values as numbers on a line that makes the Gradual part of the GLA possible, as we will now see in the next section.

1.2 How the GLA learns a grammar

Like BCD, the GLA is an error-driven online-learner: it notices when its current grammar produces a loser form, different from the ambient winner, and reacts to such an error by re-ranking constraints. However, the GLA's procedure of re-ranking is very different from the BCD one. Rather than using the Ws and Ls of winner-loser pairs as a starting point to find a new ranking, the GLA merely promotes *all* constraints that assign a W and demotes *all* constraints that assign an L.[2]

[2]This particular method of choosing constraints to promote and demote is really only one of many GLAs considered in Boersma (1997) and Boersma and Hayes (2001). However, this is the one that these authors

To illustrate: suppose a GLA learner whose current ranking values are those in 2) encounters the target form [A], and feeds /A/ to EVAL using the one-time ranking as in 3). This ranking will create the error in 5)5) below:

5)a) *An error caused by one run of the grammar in 3):*

winner~ loser	*A	*B	Ident-B	*C	Ident-A
A ~ C	L			W	W

In response, the GLA will now adjust ranking values accordingly; this process is called a learning trial:

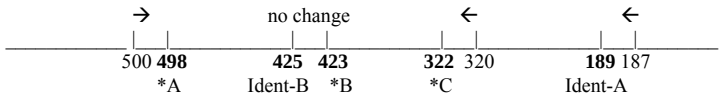
5)b) *The GLA's response to the error*

winner~ loser	*A	*B	Ident-B	*C	Ident-A
A ~ C	L →	(no change)	(no change)	←W	← W

How much are constraints promoted and demoted? Unlike in BCD, this is a question that must be answered, because our constraints are ranked by absolute values, not just relative to each other. The amount by which each constraint is moved in response to an error is referred to as the learner's *plasticity*, and the GLA assumption is that over time the learner's plasticity decreases, so that constraints move less and less in response to errors. If for example the learner's plasticity is currently 2, the actual re-ranking effect of 5)a) applied to the old grammar from 2) will be as in 6) below – here, the previous ranking values are in regular font, and the new values are in bold:

find works best – in particular, Boersma and Hayes (2001) diagnose this brand of GLA as the only one that produces the variation patterns they attempt to model -- and it's also the default version of the GLA used in OTSoft (see later this section). Therefore, I will refer to this re-ranking algorithm as "the GLA" from here onwards.

6) The new GLA grammar:



1.2.1 The (limited) power of an error in the GLA

The GLA learner does not attempt to *resolve* errors in any immediate way: the grammar in 6) is only very slightly less likely to make the error in 5a) as the previous grammar in 2) was. To remember how different this is from the BCD approach I adopted in previous chapters: suppose we had added the error in 5a) to our BCD Support instead:

7) *The initial ranking from 4):*
*A >> *B >> Ident-B >> *C >> Ident-A

8) *The error from 5a)*

winner~ loser	*A	*B	Ident-B	*C	Ident-A
A ~ C	L			W	W

9) *BCD learning result:*
*B, *C >> *A >> Ident-A, Ident-B

In the BCD approach, this error has been enough to *completely* re-arrange the grammar (compare 7) to 9), whereas the GLA learner has only slightly revised its ranking values.

Recall, however, that the BCD learner is not doomed if this re-ranking is wrong: since this error is stored in the Support, later re-rankings can undo any of these rankings if necessary. Not so in the GLA: the GLA learner does *not* store its errors for any later

use. After the learner has gotten to the ranking values in 6), it erases 5a) from its memory and starts creating new errors with its new ranking.

Over time, constraints that frequently assign Ls will move towards the bottom of the hierarchy and those that frequently assign Ws will move towards the top. In this way, the GLA learner demonstrates both intermediate stages and fluid grammatical variation. If the current grammar consistently produces errors where markedness assigns an L and faithfulness assign a W, the ranking values for M and F will approach each other, cross over, and finally move away from each other. As a result, the learner's outputs will gradually shift from the M >> F grammar to the F >> M grammar, with variation between the two along the way:

10) *Demonstrating gradual learning in the GLA*

a) An error:

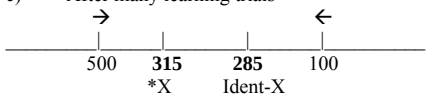
winner ~ loser	*X	Ident-X
X ~ Y	L	W

b) Initial ranking values:

The grammar's output
/X/ → [X] almost never
/X/ → [Y] almost always

Classic OT analog: **X>> Y**

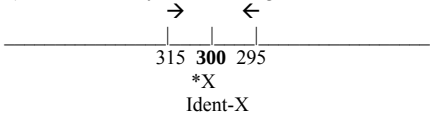
c) After many learning trials



The grammar's output
/X/ → [X] occasionally
/X/ → [Y] usually

Classic OT analog: **variation**
X>> Y, Y >> X

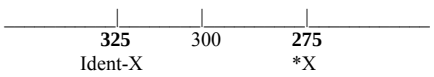
d) After many more learning trials



The grammar's output
/X/ → [X] half the time
/X/ → [Y] the other half of the time

Classic OT analog: **variation**
X>> Y, Y >> X

e) Finally, after even more learning trials



The grammar's output
/X/ → [X] almost always
/X/ → [Y] almost never

Classic OT analog: **Y >> X**

And now that /X/ is being faithfully mapped to itself almost always, the grammar is (almost) not making errors anymore (practically speaking)[3], so learning is no longer being triggered, and we have reached the final grammar.

As the example in 10) has shown, the only way to see how the GLA learns is to give it lots and lots of learning trials and track its progress over time. This is best done using a computer simulation; in this chapter I will use the simulation in OTSoft 2.1 (Hayes, Tesar and Zuraw, 2003.)[4] OTSoft takes an initial set of ranking values[5] and a table of input learning data that essentially represent ERC rows (i.e. winners, losers, and constraint violations), feeds the GLA a specified number of learning trials picked from the ERC rows, and then returns a set of new ranking values. As we will see below,

[3] For one constraint to outrank another in this stochastic OT model, practically speaking, means that their ranking values are far enough apart that choosing a ranking where their values are reversed is very, very, very unlikely. It is however true that normal distributions are asymptotic to zero: so in principle no set of ranking values makes any constraint reversal truly impossible, just very unlikely. This technicality will not concern us here.

[4] The other major GLA software comes with Praat (Boersma and Weenink, 2006),.

[5] More on the initial ranking values I used below.

choosing different numbers of learning trials allows a window into the various stages that the GLA passes through along the way from initial to final states.

1.3 Goals and core properties of the GLA

The GLA was designed to be inherently stage-like in its learning. From the present perspective, the way the GLA achieves these stages also seems very reasonable: chapter 2 section 4 provided some compelling evidence that frequency of markedness violation is a good predictor of order of acquisition, and the GLA's method of demoting L-preferring constraints precisely encodes this correlation (more on this connection in section 2 below.) Relatedly, the GLA's use of stochastic constraint rankings allows the model to learn grammars with variation, in intermediate stages of acquisition and the end state. The GLA is also designed to be robust in the face of misleading learning data, such as slips of the tongue overheard by the learner, because (unlike classic BCD) no particular error makes much of a difference (though c.f. the ESL proposal in chapter 3 section 4.4.)

Two central properties of the GLA should be kept in mind throughout the rest of this chapter. First: the GLA is *inherently* about numerical rather than ordinal OT; the stochastic view of constraints is required to make "move M1 down *a little bit*" a coherent notion. Second: a fundamental difference between GLA and BCD learning is that the GLA doesn't save its errors. In the BCD view, the Support is a record of why the current grammar has deviated from its ranking biases in the way that it has, while rankings come and go. In the contrasting GLA view, the ranking values are themselves the repository of learned information; rankings change slowly, and they require the evidence of many

errors to be reversed. The hope is that this gradualness will be enough to ensure that incorrect re-rankings are prevented.

1.4 Ranking Biases and the GLA

How can the GLA incorporate ranking biases? In one sense it is easy. Any strictly initial state bias, like the Smolensky (1996) M >> F bias, can be mimicked in this system just by giving different initial ranking values to classes of constraints: setting e.g. all Markedness constraints to the initial ranking value 500 and all Faithfulness constraints to 100.

With respect to the GLA's need for this bias: Curtin and Zuraw (2001) point out that their learner eventually finds the final grammar without the M >> F bias (i.e. with constraints all beginning with the identical ranking value), but that its early stages fluctuate wildly in ways that don't match the attested data they are trying to model. And eventual convergence is certainly not guaranteed with just *any* distribution of initial ranking values for markedness and faithfulness. If in the target grammar M1 >> F1, but at the initial state they are ranked F1 >> M1, the learner will not make an errors showing that they need re-ranking – as we have seen many times, high-ranking F means no errors in phonotactic learning. So these two constraints will only get re-ranked appropriately if they conflict with *other constraints* that cause errors where M1 assigns a W.[6]

What the next two sections demonstrate is that in addition, the bias for F-context subsets from chapter 1 is also necessary for the GLA to reach the target grammar.[7]

[6] Since, as we've seen, IO-faithfulness can assign no Ls in phonotactic learning.
[7] So far as I know, only Hayes and Londe (2006) have made this point – I return to their approach using *a priori* rankings in sections 2.5 and 6.2 below.

1.5 Chapter Roadmap

Sections 2 and 3 focus on ranking biases in the GLA. Section 2 deals with end-state grammars and the bias' role in ensuring restrictiveness; section 3 returns to the Specific-F type of intermediate stages and demonstrates the GLA's difficulty in predicting such stages without the bias. In section 4 I discuss the ways in which persistent and/or contingent ranking biases might be imposed in the GLA, and the difficulties that these approaches face. In section 5, I return to the GLA's lack of stored errors, and the problems for restrictiveness and convergence that the GLA's brand of memory-less learning causes. In section 6, I compare and contrast the GLA and BCD learners with respect to two problems in later phonological learning: exceptionality and variation in end-state grammars. As with the previous ones, this chapter's discussion will highlight the learner's need for error memory, in a format like the Support.

2. Restrictiveness and specific-to-general faithfulness relations in the GLA

I illustrate in this section the fairly simple point that the GLA's reliance on frequency of violation makes it learn superset grammars as soon as specific faithfulness constraints are admitted into CON. To do so, I report a GLA simulation using hypothetical data, stopping every few 100 trials to see the learner's progress until an stable end-state grammar has been reached. The results also show how the GLA puts faithfulness constraints on the same stringency scale into general >> specific rankings – although whether this result is a problem for the end-state grammar is not clear.

2.1. The exemplifying grammar

The hypothetical language I will use in this illustration, imaginatively called L, has two sets of vowels that are assumed to be marked: mid vowels and front rounded vowels. In L, the mid vowels [e, o] are restricted to stressed syllables only, while the front round vowels [y, ø] occur in both stressed and unstressed contexts. Both of these properties are well-attested: the former in Southern. dialects of Italian (Maiden, 1995; Flemming, 2001) and Russian (Halle, 1959, Flemming, 2001)*;* the latter in e.g. French and Turkish.

To let us build some words and make errors: I will assume that the entire vowel inventory is a 7 vowel system of the form: [i, y, u, e, œ, o, a], and that stress is always word-initial. According to the two prohibitions above: mid vowels [e, o, œ] only appear in stressed environments, and the front rounded vowel [y] appears both stressed and unstressed.

Given these parameters, the possible words that our learner must therefore learn
to produce are as in 11):

11) *Possible words of L, wrt mid/round vowels*

(a)	[képa], [kópi]	(stressed mid vowel)
(b)	[pœ́ki]	(stressed mid, front/round vowel)
(c)	[lýpi]	(stressed front/round vowel)
(d)	[píty]	(unstressed front/round vowel)

In order to learn a restrictive grammar, what we want our learners to realize is this other fact:

12) *Impossible words of L*
(a) *[kípe], *[pákœ] (no unstressed mid vowels)

To get this grammar, we first need general faith to vowel frontness and rounding to rank above *front/round , as in (13)a). To get the distribution of mid vowels, we need *[mid] to rank below the positional faith constraint, Ident-mid(σ'), but above general faith to [mid], as per (13)b):

13) *The necessary rankings of L:*

(a) Ident-rd(Seg), Ident-back(Seg) >> *front/round
(b) Ident-mid(σ') >> *mid >> Ident-mid(Seg)

Concentrating on just this portion of the grammar: the two driving forces in learning will of course be the two markedness constraints *front/round and *mid. To conflict with these markedness constraints, I will assume four Ident families of constraints – Id-Mid, Round, Front and Back – each with both a general and stressed-syllable version.

2.2 The GLA's learning input

To get started, we want to understand the errors that our learner will be making and learning from. Additionally, we need to see how the constraints that the GLA will be assuming work in treating the errors' winners. I assume that our GLA learner will be given a M >> F bias, so that *mid and *front/rd begin with ranking values of 500 and all Ident constraints at 100.

The four tableaux below present the initial errors that our learner will make. Two notes: first, I have included all seven vowel candidates in each case only to show clearly what I gave the GLA to teach it the appropriate constraint violations – candidates with equal or greater markedness than the inputs are shaded out to slightly illuminate the interesting candidates. Second: the first three tableaux deal with stressed vowels – in these cases, Ident(Seg) and Ident-σ' get the same violations, so I have collapsed them into one. In each case, the two stars Ident receives should be understood as one violation of the specific constraint, and one of the general.

As it turns out: given this constraint set, all marked vowels will map in the initial grammar to high unrounded ones (either [i] or [u], depending on the input value for front/back.) We will see this for each vowel in turn.

First: /ˈkepa/ has a stressed mid vowel; the best way to repair with this constraint set is to raise it to the high vowel [i], as all other vowels are more unfaithful (i, iii, iv) or more marked (v-vii):

14) *The tableau of violations for* /kepa/

/kepa/	*mid	*front/rd	Id(mid)	Id(bk)	Id(rd)
(i) ˈkepa	*!				
(ii) ☞ ˈkipa			**		
(iii) ˈkapa			**	**!	
(iv) ˈkupa			**	**!	**
(v) ˈkypa		*!	**		**
(vi) ˈkœpa	*!	*!			**
(vii) ˈkopa	*!			**	**!

(Note that given that all the faithfulness constraints being discussed here are symmetric: forms with the back stressed mid vowel [o] get the exact same treatment, raising to u).

The form /pœki/ has a stressed vowel that is both mid and also front-rounded; among the vowels that do not violate either markedness constraint (ii-iv), the best repairs are either of the high vowels, [i] or [u]:

15) *The tableau for* / pœki /

/pœki/	*mid	*front/rd	Id(mid)	Id(bk)	Id(rd)
(i) ˈpœki	*!	*!			
(ii) ☞ˈpuki			**	**	
(iii) ˈpaki			**	**	**!
(iv) ☞ ˈpiki			**		**
(v) ˈpyki		*!	**		
(vi) ˈpeki	*!				**
(viii) ˈpoki	*!			**	

The form /ˈlypi/ has a stressed front-rounded vowel; again, among the unmarked vowels the best repair is unrounded [i] or rounded [u]:

16) *The tableau for* /lypi /

/lypi/	*mid	*front/rd	Id(mid)	Id(bk)	Id(rd)
(i) ˈlypi		*!			
(ii) ☞ ˈlipi					**
(iii) ☞ ˈlupi				**	
(iv) ˈlapi				**	**!
(v) ˈlepi	*!		**		**
(vi) ˈlopi	*!		**	**	
(vii) ˈlœpi	*!	*!	**		

Finally, the last form /ˈpity/ has a round vowel in an unstressed context, meaning that only the general Ident faith constraints are relevant to controlling its repair. Other than that, however, its violations are the same as for the /y/ in the stressed context:

17) *The tableau for* /pity/

/pity/	*mid	*front/rd	Id(mid) Seg	Id(bk) Seg	Id(rd) Seg
(i) ˈpity		*!			
(ii) ☞ˈpiti					*
(iii) ☞ˈpitu				*	
(iv) ˈpita				*	*!
(v) ˈpite	*!		*		*
(vi) ˈpito	*!		*	*	
(vii) ˈpitœ	*!	*!	*		

As with the BCD: the goal of the GLA is get from this state to one where all inputs map to the winners (rather than to these losers or any subsequent ones.)

2.3 The stages of GLA learning

2.3.1 The initial stage

The GLA was given an initial M >> F bias of 500 >> 100, so the initial ranking values for all of our mini-grammar's constraints were as in 18) below. Note that I have listed the constraints not according to their initial ranking but instead alongside the constraints they are in conflict with:

18) *Initial ranking values*

******mid*	Id(*mid*) [Seg]	Id(*mid*) [σ'**]**	***fr/rd**	Id(**rd**) [Seg]	Id(**rd**) [σ'**]**	Id(**bk**) [Seg]	Id(**bk**) [σ'**]**

500	100	100	500	100	100	100	100

2.3.2 The intermediate stages

I ran the GLA through many different learning cycles, to get a sense of the stages that this learner tends to pass through. In each trial, I adopted OTSoft's default re-ranking plasticities, (beginning at 2, and ending at 0.002). The first real re-ranking of markedness and faithfulness occurs after about 700 trials, at which point ranking values are typically as below:

19) *Some ranking values after 700 trials*

	**mid*	Id(*mid*) [Seg]	Id(*mid*) [σ'**]**	Id(**rd**) [Seg]	Id(**bk**) [Seg**]**	***fr/rd**	Id(**rd**) [σ']	Id(**bk**) [σ'**]**
(a)	320.0	287.0	287.0	234.4	235.0	230.6	166.8	168.2
(b)	339.1	260.9	260.9	234.2	236.7	229.1	189.2	203.1
(c)	304.7	295.3	295.3	238	234	228	218	166

The crucial change in this grammar is that *front/rd is now ranked equally with or below general Id-rd and general Id-back. The output distributions table below in 20) shows that this re-ranking has increased the number of options for treating front rounded vowels, and reorganized their frequency:

20) *The output distributions after 700 trials*[8]

target	output	(a)	(b)	(c)
(i) ˈkepa	ˈkipa	100%	100%	99.8%
	ˈkepa			> 1%
(ii) ˈpœki	ˈpuki	5%	> 1%	1.4%
	ˈpyki	9.1%	4%	
	ˈpyki	85.9%	95.8%	98.4%
	ˈpœki			> 1%
(iii) ˈlypi	ˈlupi	> 1%	> 1%	1.4%
	ˈlipi	9.1%	4%	
	ˈlypi	85.9%	95.8%	98.6%
(iv) ˈpity	ˈpitu	> 1%	> 1%	1.4%
	ˈpiti	9.1%	4%	
	ˈpity	85.9%	95.8%	98.6%

In BCD terms, the ranking that this stage roughly matches is:

21) *mid >> Id(rd), Id(bk) >> *front/rd >> Id(rd)-σ', Id(bk)-σ', Id(mid)-both

Unsurprisingly, it is the general Id-round and back constraints that have gotten above Id-round(σ'), because they assign more Ws in (19). And note again that both Id-mid constraints have the same ranking value in every run.

After about 1000 trials, the learner has gotten to a third stage of ranking:

22) *Some ranking values after 1000 trials*

	Id(*mid*) [Seg]	Id(*mid*) [σ'**]**	**mid*	Id(**rd**) [Seg]	Id(**bk**) [Seg**]**	***fr/rd**	Id(**rd**) [σ']	Id(**bk**) [σ'**]**
(a)	302	302	298	236	236	228	176	162
(b)	302	302	298	238	234	228	186	180
(c)	302	302	298	236	234	230	200	192

[8] At the end of each run of the GLA, OTSoft tested its final state grammar with a 1000 trials on each input. The percentages I give in these output distribution tables reflect the results of those tests.

The crucial change at this stage is that *mid has gotten far enough below Faith that mid vowels are no longer being made high. Along with the continued demotion of *front/round, the learner has now reached a state where very few errors are being made:

23) *The output distributions after 1000 trials*

target	output	(a)	(b)	(c)
(i) ˈkepa	ˈkipa	2.9%	2%	2.7%
	ˈkepa	97.1%	98%	97.3%
(ii) ˈpœki	ˈpuki	> 1%	1.9%	1.4%
	ˈpyki	> 1%		7.5%
	ˈpyki	2.9%	2%	2.1%
	ˈpœki	96.7%	96.1%	89%
(iii) ˈlypi	ˈlupi	> 1%		8%
	ˈlipi	> 1%	1.9%	0.9%
	ˈlypi	99.6%	98.1%	91.1
(iv) ˈpity	ˈpitu	> 1%		8%
	ˈpiti	> 1%	1.9%	0.9%
	ˈpity	99.6%	98.1%	91.1

2.3.3 The end-state grammar

For this learner, the third stage at 1000 trials is pretty much the end-state ranking; none of the crucial rankings that its values embody are going to get revised. To make this perfectly clear:

24) *After 50,000 trials*

	Id(*mid*) [Seg]	Id(*mid*) [σ'**]**	*****mid*	Id(**rd**) [Seg]	Id(**bk**) [Seg**]**	***fr/rd**	Id(**rd**) [σ']	Id(**bk**) [σ'**]**
(a)	304	304	296	238	238	224	202	144
(b)	304	304	296	238	236	226	216	152
(c)	304.1	304.1	295.9	238	236	225.4	212	196

25) *The output distributions after 1000 trials*

target	output	(a)	(b)	(c)
(i) ˈkepa	ˈkepa	100%	100%	100%
(ii) ˈpœki	ˈpœki	100%	100%	100%
(iii) ˈlypi	ˈlypi	100%	100%	100%
(iv) ˈpity	ˈpity	100%	100%	100%

Since no more errors are being made, these ranking values are therefore final. The comparable BCD final grammar is one that includes these two rankings:

26) a) Id(mid)-both >> *mid
b) Id(rd), Id(bk) >> * front/mid >> Id(rd)-σ', Id(bk)-σ'

2.4 Summarizing the results

2.4.1 The superset grammar: mid vowels

The most important thing we've seen is that *crucial* stringency relations between faithfulness constraints cause the GLA to choose superset grammars. When presented data like 27), the GLA and (my) BCD algorithms get the two different rankings in 28):

27)

winner ~ loser	*mid	Id(mid) [Seg]	Id(mid) [σ'**]**
képa~kípa	L	W	W

28)a stabilized GLA ranking: Id(mid), Id(mid)-σ' >> *mid
29)b my BCD ranking: Id(mid)-σ' >> *mid >> Id(mid)

The problem with the GLA-acquired ranking is that it assumes mid vowels are permitted *everywhere,* having only observed them in the stressed syllable context. Thus

the GLA learner will faithfully parse an input with an unstressed mid vowel like [kíp**e**] – precisely the kind of form we wanted our learner to rule out:

29) *The over-generation of the GLA ranking*

/kípe/	Id(mid)	Id(mid)-σ'	*mid
(i) ☞ kípe			*
(ii) kípi	*!		

2.4.2 The 'Anti-Paninian 'Ranking: front rounded vowels

A second difference between the rankings that this simulation demonstrated is that errors like 31) below give different rankings from the BCD learner as well:

30)

winner ~ loser	*front/ rd	Id(bk) [Seg]	Id(bk) [σ'**]**	Id(rd) [Seg]	Id(rd) [σ'**]**
lýpi~lípi	L	W	W	W	W
píty~píti	L	W		W	

31)a stabilized GLA ranking: Id(rd), Id(bk) >> *rd >> Id(rd)-σ', Id(bk)-σ'
33)b my BCD ranking: Id(rd)-σ', Id(bk)-σ' >> Id(rd), Id(bk) >> *rd

The difference between these rankings is the position of the positional faithfulness constraints: BCD will install them high when it can, where the GLA doesn't promote them any higher than it has to.

It is not immediately clear which of these end-state grammars should be preferred – because without alternations, it is not clear how we could tell whether adults learning such a language choose one ranking over the other. However, the GLA's quick

promotion of general faithfulness constraints also has interesting but not altogether helpful consequences for its *stages* of acquistion. This is the focus of the next section.

3. Intermediate stages and the Specific-F >> General-F bias in the GLA

A series of studies – Curtin and Zuraw (2001), Boesrma and Levelt (2000); Levelt and van der Vijver (2004) – have proposed using the GLA to generate the intermediate, often overlapping stages seen in children's developing grammars, specifically with reference to the Fikkert/Levelt corpus of Dutch phonological acquisition.

First: Levelt and colleagues connect the order of syllable shape acquisition in Dutch to the lexical frequency of syllable types, and show how the GLA can therefore model these acquisition stages. The constraint set they use – while good for illustrating the GLA and their point – is rather idealized: in particular, their grammar has a monolithic general faithfulness constraint interposed with syllable-shape markedness constraints of various degrees of specificity:

32) *Constraints from Levelt and van der Vijver*
Markedness: Onset, NoCoda, *ComplexOnset, *ComplexCoda
Faithfulness: Faith

3.1 The Specific F stages that require the ranking bias

As we saw in the previous section, however, a more articulated set of faithfulness constraints causes problems for the GLA model. The frequency of faithfulness violations always gets a *general* faithfulness constraint promoted either as fast or faster as anything specific to it. As a result, the GLA does not predict intermediate Specific-F stages like the

ones discussed in chapter 3, sections 1.3 and 2.3. Substituting complex onsets for the round vowels in the simulation above, this means that the GLA does not predict the intermediate French stages found by Rose (2000) and Kehoe and Hilaire-Debove (2003) – this is schematized again below:

33) *How the GLA misses the Specific F stage of French complex onsets*

a) *Initial state*: *ComplexOnset >> Max(Seg)-(σ'), Max(Seg)
Observed intermediate state: Max(Seg)-(σ') >> *ComplexOnset >> Max(Seg)
Target state: Max(Seg) >> *ComplexOnset

b) *Observed winners*: bá blá bablá blabá

c) *GLA Constraint movement, created by errors at initial state:*
*ComplexOnset: demoted by every word with *any* complex onset, i.e:
blá bablá blabá

Max-Seg: promoted by every word with *any* complex onset, i.e.:
blá bablá blabá

Max-Seg(σ'): promoted by every word with *a stressed* complex onset, i.e:
blá bablá

d) *Upshot:* *ComplexOnset and Max(Seg) fall and rise at the same rate, while Max(Seg)-σ' rises slower

e) *The GLA's first new stage:*
Max(Seg) >> *ComplexOnset >> Max(Seg)-σ': …saving *all* complex onsets

f) *The mismatch between (e)'s ranking and the observed intermediate stage:*

	/blablá/	Max(Seg)	*ComplexOnset	Max(Seg)
	(i) babá	**!		**
intermediate stage winner	(ii) bablá	*!	*	*
GLA's winner	(ii) ☞ blablá		**	

3.2 The Specific F stages that don't require the bias: Curtin and Zuraw (2001)

An interesting result, however, is that GLA *can* create intermediate stages in which specific-faithfulness constraints play a role, even though they rank below General-Faith. The Curtin and Zuraw (2001) simulation of Dutch syllable truncation provides such an example.[9]

In their simulation, Curtin and Zuraw provided a GLA learner with schematic lexical items (meaning 2-to-4 syllable strings with one main stress each), at frequencies that approximate the CELEX-based Dutch lexicon. Below I list the Markedness and Faithfulness constraints they used, and their stringency relations: [10]

34) *Curtin and Zuraw (2001) constraints*

(a)

Markedness	Faithfulness
All-Ft-L	Max-PitchProm (=Max-Stress-σ)
All-σ-L	Max-FinalProm (=Max-Final-σ)
Parse-Syll	Max-σ
FtBin[11]	

(b)

M-stringency	F-stringency
All-Ft-L	Max-Stress-σ and Max-Final-σ
is less stringent than	*is less stringent than*
All-σ-L	Max-σ

In keeping with the M >> F bias (which they point out was crucial to get their early stages to look realistic), they started all M and F constraints with ranking values of 500 and 100 respectively. The general result was that, as in Fikkert's diary study, this simulated learner went through four stages:

[9] My thanks to Kie Zuraw for discussing this data when I didn't understand what I was saying about it yet.
[10] Readers may remember that I discussed this intermediate stage in chapter 3 section 4.3, without using the constraint All-σ-L.
[11] Curtin and Zuraw actually use the constraint 'FootMax', which says "Feet are maximally disyllabic."

35) Stage 1: Truncation to one syllable
Stage 2: Truncation to two syllables (one foot)
Stage 3: Necessarily one or two feet (no syllables unparsed)
Stage 4: Unfooted syllables also allowed

Comparing the constraint relationships in 34), and the rankings for these stages, the GLA did precisely what we've seen it does: demoted the general Markedness constraints fastest, and also the promoted the general Faith constraints. Given what I have said above about the GLA's over-zealous promotion of general faithfulness, it is therefore important to note that this learner's stage 2 – truncation to a single foot – *did* preserve syllables in privileged positions (stressed and final syllables).

To see why, I show Curtin and Zuraw's first re-ranking – stage 2 – in 36) below. As expected, the most general markedness constraint, All-σ-L, has gotten below the most general faithfulness constraint, Max-σ:

36)a FtBin, Parse-σ, All-Ft-L >> Max-σ >> All-σ-L >> Max-stressed-σ, Max-finalσ

Together, the top three Markedness constraints ensure that every output is no bigger than a disyllabic foot; I collapse all three into the single constraint "OneFoot":

36)b "OneFoot" >> Max-σ >> All-σ-L >> Max-stressedσ (and final-σ)

Given that OneFoot is undominated – outputs are going to be no bigger than two syllables, so any input with more than two input syllables must violate general Max-σ. In such a case, the specific Max constraints play the tie-breaking roles even though they are low-ranking:

37) *The GLA-style stage 2: the role of undominated Markedness*

/$S_1W_2W_3$/	"One Foot"	Max-σ	All-σ-L	Max-stressed-σ	Max-final-σ
(i) (S_1W_2)W_3	*!		***		
(ii) (S_1W_2)		*	*		*!
(iii) ☞ (S_1W_3)		*	*		
(iv) (S_2W_3)		*	*	*!	

In this ranking, it doesn't really matter that All-σ-L is between the general and specific faithfulness constraints[12]. The respective ranking of specific and general Max is moot in cases where higher-ranked constraints force violation of the general faith constraint.

This analysis of Curtin and Zuraw's stage 2 lets us see why the GLA does not extend to all Specific-F stages. Returning to the example of French complex onsets discussed in the previous section: the problem there is the lack of an undominated Markedness constraint to play the role of 'OneFoot'. The only Markedness at hand requires *no* complex onsets, and so the Specific-F >> M >> General-F ranking is crucial to protect those in stressed syllables.

38) *The intermediate French stage (repeated)*

/blablá /	Max-Seg (Stressed-σ)	*Comp Ons	Max-Seg
(i) blablá		**!	
(ii) blabá		*	*!
(iii) ☞bablá		*	*
(iv) babá	*!		*

[12] This is what makes this ranking different from the Anti-Paninian rankings Prince talks about, where it *is* crucial for other ranking reasons that General-Faith >> M >> Specific-Faith be true. See also chapter 2 section 7.3 for discussion of acquiring true Anti-Paninian rankings.

To get a GLA-style Specific-F stage here, we would need an undominated constraint that wanted *no more than one* complex onset in every word, so that the specific faith constraint could choose its placement:

39) *The GLA version of the intermediate French stage, with a spurious constraint*

/blablá /	***OCP-complex onset***	Max-Seg	*Comp Ons	Max-Seg (Stressed-σ)
(i) blablá	*!		**	
(ii) blabá		*	*	*!
(iii) ☞bablá		*	*	
(iv) babá		**!		*

So far as I know, such an OCP constraint is not empirically supported. But the larger point here is that deriving Specific F stages in GLA learning relies on the existence of independent markedness pressures to make most of the ranking decisions. This requirement will surely not be met for all such attested stages, so the GLA will not be able to derive them all.

3.3 Interim Summary

Sections 2 and 3 demonstrated that a Specific >> General Faithfulness bias remains necessary in the GLA. The next section considers how that bias should or could be implemented.

4. Persistent biases, contingent biases, and the GLA

The need for the specific >> general faithfulness bias in GLA learning has also been noted recently by Hayes and Londe (2006). In part of their learning simulation of Hungarian vowel harmony, they use a GLA algorithm which includes a ranking bias for

high-ranking IO-Ident[back]-*Root* >> IO-Ident[back]. They point out in a footnote that general faith climbs too high without such a bias.

In the BCD discussion of chapter 1, ranking biases came in two flavours. First were what I call "definitional" biases, whose effects can be read off the constraints themselves: Markedness >> Faith, OO-Faith >> IO-Faith, and Specific-Faith >> General-Faith (of the language-independent nature.) There are at least two ways in which the GLA could incorporate persistent ranking biases of the definitional sort.

One option would be to assign different plasticities according to both constraint type and direction of re-ranking. To enforce a continued preference for ranking A >> B, we would promote A a lot when it prefers winners, but only demote it a little when it prefers losers, and we do the reverse for B (promote a little but demote a lot.)[13]

A more direct way is what Hayes (200X)'s OTSoft manual calls A Priori Rankings. Hayes describes the OTSoft implementation of a priori rankings as follows:

40) "OTSoft implements a priori rankings for the Gradual Learning Algorithm as follows: it minimally adjusts the initial ranking values so that any two constraints that are ranked a priori are at least *x* units apart, for some value of *x*. Then, as it incrementally adjusts the ranking values of the constraints, it monitors the a priori rankings so that they continue to be enforced by at least a distance of *x* ranking values or greater. The default setting of *x* is 20, which is very close probabilistically to being an obligatory ranking." (Hayes, 200X: 21)

Imposed in this way, a priori rankings have the effect of moving constraints *independent of the learning data* in the case that another constraint is getting too close. This approach can thus create a persistent ranking bias for any definitional ranking bias we like.

[13] Thanks to Joe Pater for discussion of this potential approach.

However, chapter 1 was also concerned with what might be called "calculable" biases – language-dependent Specific-F >> General-F, as well as Prince and Tesar's (and Hayes') principles for choosing the right IO-Faith constraint to install in each stratum. These ranking biases were in fact more like ranking *principles*: given a certain set of IO-Faith constraints that prefer a particular set of winners, they determine the safest bet for ranking in the next stratum. It is worth re-stressing that we cannot hardwire all Specific >> General-F relations into the learner because of contingent stringency, and that morphological categories can in principle be implicated in such stringency relations.

The twin problems for including calculable ranking biases in the GLA are (a) how to calculate them and (b) what ranking consequences they should have. The first problem arises because the GLA does not store its errors, so it does not have a Support to draw Context Tables from. Of course, the real-life GLA learner must be learning not just phonotactics but also a real language, and thus a lexicon – so, we could simply say that contingent faithfulness relations are calculated from all the stored URs in the lexicon (with the continued assumption of the Identity Hypothesis whereby inputs are identical to observed winner outputs.)

But there is also the larger question of what these ranking biases actually do to the GLA's re-rankings. To be concrete, we can first assume our GLA learner is equipped with A Priori rankings for all definitional specific-to-general faith relations – and further that every time our learner goes to promote any W-assigning faithfulness constraints it first consults the lexicon and checks for a contingent specificity relationship between that constraint's contexts and all others.

And if the GLA learner discovers such a specific-to-general relationship? What should it do? We could first say that it adds that relationship to its list of A Priori rankings – and so if the current error only provides evidence for promoting the less specific constraint, it should nevertheless move up the more specific one a healthy 20 points above the more general one, as instructed in 40) above.

But what if the learner has discovered this specificity relation *after* our constraints have already gotten too high? For example: imagine that our GLA learner is trying to learn Language 1 from chapter 2 §4.3.1.1, where initial syllables are a special case of stressed syllables, and only initial syllables can contain mid vowels. If both of these Ident[mid] constraints have already climbed above *mid in the ranking – it will do no good to notice at some later point that Ident[mid]-σ1 should have been ranked *a priori* above Ident[mid]-σ'. What the learner needs to do to get out of this superset grammar is to demote Ident[mid]-σ' below the relevant markedness constraint *[mid]. This is a way in which the GLA's disconnect between errors and ERC rows seems to cause it real problems: the GLA has no notion of demoting below a certain constraint – it only moves constraints along its numerical scale, without reference to the (ordinal) position of any other constraints. And as it stands, it is not clear how the GLA could incorporate the necessary reasoning into its method of constraint re-ranking.

5. A first problem with not storing errors: winner misparses

The rest of this chapter returns to the issues raised in chapter 2 about the need for stored errors (beyond any specificity calculations), and their absence in the GLA.

In some ways, memoryless-learning has its benefits: one way is in dealing with occasional noise in the learning data. Since the GLA's response to each individual learning datum is conservative: if a particular error is in some way wrong, it will have a negligible effect on the end-state ranking. And since the GLA does not remember its errors, there is no sense in which the grammar remains responsible for this error as it continues learning.

However, chapter 2 presented two learning situations in which a memory for learning errors was crucial. One problem, addressed in section 4.1 below, is that unlike occasional noise, persistently misleading learning data in the form of winner misparses *will* drive the GLA to adopt a superset grammar. Once such grammars have been learned, the GLA's lack of stored errors prevents its recovery even once the misparses are fixed (cf. the Support-based treatment of this problem in chapter 3 §5.2)

Another problem, mentioned briefly in chapter 2 §2.2, is that not storing errors prevents the GLA from seeing fundamental inconsistency between the errors it is making. One place this inconsistency will be present in natural languages is if the target grammar includes exceptionality. What I demonstrate here is that the effect of an exceptional grammar will be the same as a *variable* one – and that revising lexical entries cannot provide the whole solution to this problem. Sections 5.3-5.5 discuss this issue, including the more complex treatment of exceptionality in Zuraw (2000) and Hayes and Londe (2006).

5.1 The GLA's treatment of misparsed winners

To demonstrate the problem, I return to the example of coda devoicing and the mis-syllabification of winners discussed in chapter 2 §4.2. The core of the problem

presented there was the difficulty in learning a correct grammar of voicing neutralization in coda position, when relying on learning data whose voiced onset segments are sometimes mistakenly syllabified as codas:

41) *The target grammar (repeated from chapter 2 ex. 45)*
Ident[voice]-Onset >> ***VoicedObs >> Ident[voice]**

42)a) *Learning onset voicing with the right ERC (chapter 2 ex. 46)*

winner ~ loser	*VoicedObs	Ident[voice]-Ons	Ident[voice]
[ká.**bl**a] ~ [ká.**pl**a]	L	W	W

42b) *Learning onset voicing with the misparsed ERC (chapter 2 ex. 47)*

winner ~ loser	*VoicedObs	Ident[voice]-Ons	Ident[voice]
[ká**b.l**a ~ ká**p.l**a]	L	**e**	W

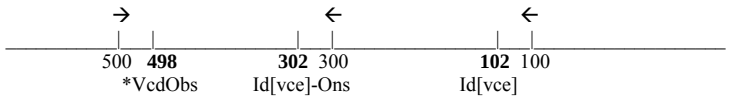
As we saw in section 2, the GLA does not correctly rank specific faithfulness constraints over general ones like Ident[voice]-Onset >> Ident[voice] without a ranking bias. Suppose therefore that we equip the GLA with an initial Specific-F >> General-F ranking bias à la Hayes and Londe (2006) (putting aside the issue of language-specific cases for the sake of argument.) In this case, we will therefore start our learner off with a ranking like 43):

43) A hypothetical GLA initial state, with both M >> F and Spec-F >> Gen-F:

The learner who knows that words like [dábla] are parsed with a medial complex onset will make errors corresponding to the correct learning ERC row in 42a). In GLA

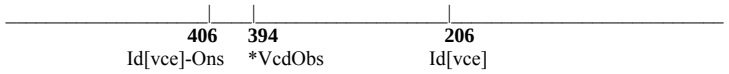
learning, this error will cause a small demotion of *Voiced Obs and a small promotion of *both* Ident[voice] constraints:

44) The re-ranking effect of the right learning ERC:



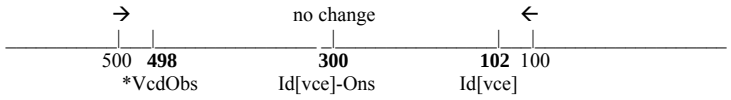
Repeated errors like 44) will eventually promote Ident-Onset *above* *VcdObs; once Ident-Onset has gotten high enough, no more errors will be made, and thus the learner will have found the right end-state grammar:

45) The right final grammar (hypothetical values):



But what about the learner who misparses the relevant winner as [ká**b.l**a], and so makes the misleading error in 42b)? The GLA will react to this error with a small demotion of *VcdObs and a small promotion of just the *general* Ident[vce] constraint:

46) The re-ranking effect of the *wrong* learning ERC:



What will this learner's end-state grammar be? For the learner who persists in their syllabification misparse of words like 'kábla', the final grammar will be the wrong one: *VcdObs will continue to sink and general Ident[vce] will continue to rise until their ranking flips and errors stop being made. With these initial ranking values (and the OTSoft default plasticity settings, as in section 2 above), this re-ranking takes about 400 trials; further refinement continues until the grammar reaches a ranking where general Faith is high enough above Markedness that no errors are being made, practically speaking:[14]

47)a) Ranking values after 400 trials
302.660 Ident[vce]
300.000 Ident[vce]-Onset
297.340 *VcdObstruent

b) Ranking values after 50,000 trials
306.000 Ident[vce]
300.000 Ident[vce]-Onset
294.000 *VcdObstruent

As we've seen, the GLA does not have a mechanism for unlearning things it has already learned. As a result, the GLA learner's only hope is to learn the right complex onset syllabification of [ká.**pl**a] before the crucial re-ranking has happened. If for example only 250 trials have gone by, the re-ranking will not yet have undone the initial rankings:

48) Ranking values after 250 trials
362.236 *VcdObs
300.000 Ident[vce]-Ons
237.764 Ident[vce]

[14] Using an A Priori ranking, rather than an initial ranking bias, will not help this learner at all – it will get Ident[vce]-Onset ranked higher than Ident[vce], but the learner still won't stop making errors until general Ident is above Markedness.

If at this point the learner started making errors using the correct winner parse, Ident[vce]-Onset will start assigning Ws, and *both* Ident constraints will now begin to climb as in 44). We can simulate this second stage of GLA learning by taking the numbers in 48) as initial ranking values, and feeding the GLA the correct input file repeated below:

49) *Learning onset voicing with the right ERC*

winner ~ loser	*VoicedObs	Ident[voice]-Ons	Ident[voice]
[ká.**bl**a] ~ [ká.**pl**a]	L	W	W

In this situation, the learner *will* get to the right end-state grammar. Since both Ident constraints are now rising at the same pace, the more specific Ident constraint will overcome *VoicedObstruent before the general constraint, and this will be enough to stop making errors. In this case: errors stop being made after about 75 trials, and the ranking stabilizes correctly:

50) a) Ranking values after 75 trials
333.264 Ident[vce]-Ons
328.972 *VcdObs
271.028 Ident[vce]

b) Rankings values after 50,000 trials
336.000 Ident[vce]-Ons
326.236 *VcdObs
273.764 Ident[vce]

What this example shows is that the GLA's robustness to persistent misparses is only a function of the *number* of misparsed errors – and, when using initial ranking biases, the speed with which the relevant constraints move in the hierarchy.

5.2 Winner misparses and markedness: the same problem

It should be noted that the misparse danger outlined above exists independent of faithfulness and its specificity relations. Suppose that CON included two markedness constraints: *Pharyngeal, and *Pharyngeal-affix, and that the learner was attempting to acquire a language in which pharyngeal consonants only occur in roots but not affixes. This means the target grammar would be as in 51):

51) *The target grammar*
*Phar-Affix >> Ident-Phar >> *Phar

With the right ERCs, learning this ranking presents no problem for the GLA. Assuming again the M >> F initial ranking values, the learner will at first make mistakes on pharyngeals and thereby create ERCs like in 52) below (<u>roots</u> are underlined):

52) The right ERC for the root-pharyngeal language

winner ~ loser	*Phar-Affix	*Phar	Ident-Phar
<u>ʔabat</u> ~ <u>gabat</u>	e	L	W

As a result only the general *Phar constraint will be demoted:

53) The GLA re-ranking resulting from 52)

Trouble will arise, however, if the learner happens to misparse words like "ʔabat" as containing the prefix "ʔa-".[15]

54) The morphological misparse ERC

winner ~ loser	*Phar-Affix	*Phar	Ident-Phar
ʔabat ~ gabat	**L**	L	W

This ERC will make the GLA demote *both* the general and the affix-specific *Phar constraint:

55) The GLA re-ranking resulting from 54)

This is the same situation as we encountered for the syllabification case in the previous section. If this representational misparse lasts long enough, the affix-specific *Phar constraint will eventually get re-ranked below Ident-Phar:

56) *The final-state grammar, learned from incorrect ERCs like 54)*
Ident-Phar >> *Phar-Affix, *Phar

[15] A couple remarks in defense of the claim that learners might misdiagnose part of a root instead as an affix. The first is that some spontaneous cases of this sort have been cited in the literature – e.g., the exchange: Parent: "Behave!" Child: "I *am* have!" A reasonable counterargument is that this misparse occurred precisely because English *has* a free-standing 'be' morpheme, whereas my *Pharyngeal-Affix example requires that the child has mistaken something for an affix when the language crucially does *not* have one. While I do not know of any strictly comparable cases in the literature on morphological acquisition, the most plausible scenario I can think of comes from a language like Hebrew, with both non-concatenative morphology as well as prefixes. The suggestion would be that the non-concatenative properties make children sensitive to vowel patterns, independent of their associated consonants, and that this could transfer over to prefix recognition. Thus, the learner might mistakenly decide on the basis of a series of prefixes and clitic-like things that prefixes were of the format [Cɛ] (e.g. Hebrew prepositions [bɛ-/lɛ-/kɛ-/mɛ- as well as verbal pronouns tɛ-/yɛ-/nɛ-). So when faced with a root that began [ʔɛ…] (in Hebrew prime that contained such pharyngeal-initial roots, that is), the learner could choose the relevant morphological misparse.

And even once the learner has learned the correct parse for "ʔabat", and determined that it does not contain an affix – it is too late for the GLA learner to do anything. Errors are no longer being made, so there is no error-driven incentive to re-rank any of these constraints. And without a memory for its errors, the GLA learner has no way to wonder whether alternative explanations for marked structures like pharyngeals are available.

6. Exceptions and end-state variation

This section looks beyond the stage of pure phonotactic learning, to the later stages where the Identity Hypothesis is abandoned and the learner has adopted unfaithful mappings between inputs and outputs. The goal here is not to explicate how unfaithful inputs are learned, but to consider the behaviour of the BCD and GLA learners once these correct, unfaithful inputs have been chosen.

Among the various grammatical patterns the learner must cope with, I focus on grammars that encode exceptionality and variation. These two patterns are discussed in tandem here because learning theories of both numerical and ordinal OT have sometimes conflated them, though not necessarily to positive effect.

The upshot of this discussion is that the GLA is inherently better suited to handling variation – as is well known – but that the BCD learner is in fact better suited to learning exceptionality. I therefore discuss a way that the BCD could be used to learn variation (extending work by Pater, to appear), and also discuss how the learner of Hayes and Londe (2006), which relies in part on the GLA, would learn exceptions through a non-GLA mechanism. I also emphasize how any learner's success in discovering the

right patterns of exceptions or variation will derive from their ability to compare stored errors in something like the Support.

6.1 The GLA's treatment of exceptionality

The Gradual Learning Algorithm's approach to variation in phonological development is to make variation an inherent property of the OT grammatical system. Because constraints can overlap in their distribution to varying degrees, the GLA can build a grammar that produces variation between different optimal outputs with probabilities that closely match the attested frequency of variants. The Ilokano example of Boersma and Hayes (2001) shows that the GLA can mirror cases of free variation in this way, by getting just the right constraints ranked in a clump.

A necessary difference between grammars, however, is the split between free variation and grammatical *exceptions,* which show similar patterning among outputs but require different constraint rankings.[16] This difference is not something the GLA is equipped to detect. Without storing errors, the learner can't know whether they have evidence for one pattern of variable grammatical behaviour, or for two different patterns of categorical grammatical behaviour.

What I demonstrate in the rest of section 6.1 are two languages with different input-output mappings, that will nevertheless teach a simple GLA the same variation grammar in both cases. I will then return in §6.2 to more recent GLA on exceptions – the proposals of Zuraw (2000) and Hayes and Londe (2006), which use the GLA in more nuanced ways and do learn exceptionality.

[16] See Pater (to appear), p. 6, for explicit discussion of this difference.

6.1.1 Two languages and their codas

The first example language is the Variable Coda language: one in which every coda in the lexicon is variably deleted in a way controlled by the grammar. To simplify from the real-life grammatical conditions of variable coda deletion attested in natural languages (segmental context, stress, word position, etc.), I will describe the variable coda language as treating *every* coda in citation form exactly the same: retaining it two thirds of the time, and deleting it the other third. This is the grammar that the GLA will learn correctly:

57) *The Variable Coda language*

a) the lexicon (inputs)	b) the outputs
/pak/	[pak], 67% of the time; [pa] 33%
/blag/	[blag] 67% of the time, [bla] 33%
/gri/	[gri] 100%
/tro/	[tro] 100%

In contrast, we might also have an Exceptional Coda language: in this case, most lexical items with codas retain them on the surface in citation form, but an exceptional class of lexical items lose them.[17] (To support the claim that these exceptional words have underlying codas, we can say that they do surface in some non-citation environments – e.g., when preceding a vowel-initial word.)

Unlike the previous case, the exceptional coda language varies between coda faithfulness and coda deletion on an item-by-item basis. As 58) shows, some input codas in a one-word utterance are *always* preserved, while others are *always* deleted:

[17] I call this the exceptional class only because it will be less frequent in the hypothetical lexicon. Depending on the analysis of exceptionality adopted, it could be that the lexical items that *retain* their codas would be the ones given special treatment and thus perhaps the exceptions, even though they are more numerous.

58) *The Exceptional Coda language*

a) the lexicon (inputs)	b) the outputs
/pak/	[pak]
/pa/	[pa]
/blag/	[blag]
/bla/	[bla]
/grip/ - exceptional	[gri]
/tro/	[tro]

This is the language that the GLA will ultimately treat as another instance of free variation (although c.f. Zuraw (2000), Hayes and Londe (2006) – see section §6.2.)

6.1.2 Learning the variable coda grammar

During phonotactic learning, both of these languages will look the same to the learner. Some lexical items will have codas, and some will not, and nothing else will be revealed. This is schematized below:

59)

a) *I-O mappings assumed in phonotactic learning*	b) *Real Inputs*	
	variable codas	*exceptional codas*
/pak/ → [pak]	/pak/	/pak/
/pa/ → [pa]	**/pak/**	/pa/
/blag/ → [blag]	/blag/	/blag/
/bla/ → [bla]	**/blag/**	/bla/
/gri/ → [gri]	/gri/	/**grip**/ - exceptional
/to/ → [tro]	/tro/	/tro/

To learn any of the bold inputs in 59b) above, the learner has to get past the phonotactic learning stage – a process that I have so far not dealt with in either the GLA or BCD contexts. For present purposes, I will assume that the learner has been augmented so as to learn variation among lexical inputs. This could be done, in principle, by determining that two outputs ([pak] and [pa]) have the same meaning, attempting to

collapse them into one lexical entry, and then deciding that /pak/ is the right input choice because CON includes rankings that map /pak/ to both outputs but no ranking that maps /pa/ to both (i.e. there is no constraint that prefers final k-epenthesis.)[18]

I will therefore grant that our learner has adopted the new input-output error pairs in 60) below. Since we now have more than one kind of error, the GLA's behaviour will depend in part on the frequency of their occurrence. In the Variable Coda language, every input with a coda retains it two thirds of the time, and loses it the other third. This is reflected in the frequency column that I have added into the table below; OTSoft mirrors these frequencies in generating errors to feed the GLA:

60) *New ERCs for the variable grammar (as handled by the GLA)*

correct input	winner ~ loser	frequency	NoCoda	Max	Dep
(i) /pak/	pak ~ pa	66.7%	L	W	e
(ii) /blag/	blag ~ bla	66.7%	L	W	e
(iii) **/pak/**	pa ~ pak	33.3%	W	L	e
(iv) **blag/**	bla ~ blag	33.3%	W	L	e

To model post-phonotactic learning, I gave this input data in 60) to the GLA. Here is what OTSoft turned out:

61) *Ranking values for the variable coda grammar after 50,000 trials*

new ranking	NoCoda	Max	Dep
(a)	238.2	231.8	230
(b)	238	231.6	230
(c)	240	230	229.5

[18]It is not clear to me how the GLA learner would work this out. Since the GLA does not reason about rankings in e.g. the way BCD does, I do not know how it could determine whether CON provides a stable grammar that maps /pa/ to [pak] or vice versa. To my knowledge, this is an unresolved issue.

(d)	239.2	231.6	229.2

62) *Output distributions for the variable coda grammars*

input	output	ranking (a)	ranking (b)	ranking (c)	ranking (d)
/pak/	☞ pak	74.1%	64.6%	63.3%	79.6%
	☞ pa	25.8%	35.4%	36.7%	20.4%
	paka	> 1%			

In short: the GLA has learned this variable coda grammar correctly. One could get the frequencies closer to the targets by with some fine-tuning – e.g. changing the schedule of constraint plasticities – but for our purposes it is enough to note that this ranking allows codas more often than not, and that any lexical coda is up for deletion between a fifth and a third of the time.

6.1.3 Learning the exceptional coda grammar

As in the previous section, the learner of the Exceptional Coda language must first do some morpho-phonological processing and change its lexical inputs to realize that something other than the identity map between inputs and outputs is necessary. Here, the learner must find a way from alternations to notice that some lexical items with codas nevertheless surface in citation form without them, getting to a new set of ERC rows like those below:

63) *New ERCs for the exceptional grammar*

correct input	winner ~ loser	NoCoda	Max	Dep
(i) /pak/	pak ~ pa	L	W	e
(ii) /blag/	blag ~ bla	L	W	e
(iii) **/grip/**	gri ~ grip	W	L	e

This input set was constructed so that the ratio of regular to exceptional coda forms is 2:1 – that is, two thirds of the lexicon's words with codas retain then, while the other third lose them. But recall that this language differs from the variable coda one in that no single lexical item varies: /pak/ always retains it coda in this context; /grip/ always loses its coda. And this lexical regularity is what the GLA cannot track – all it can know is that two thirds of its errors look like 63i), and that the other third look like 63iii). So when I again gave the GLA these three new errors in 63), the results looked just like those in the previous section:

64) *Ranking values for the exceptional coda grammar after 50,000 trials*

initial rankings	new ranking	Dep	Max	NoCoda
62a)	(a)	238.9	231.2	230
	(b)	238.7	231.2	230.1
62b)	(c)	238.4	231.8	229.8
	(d)	238	231.9	230

These rankings produce variable codas in *all* lexical items. The table in 65) below shows the frequency of coda retention and deletion: it is of course the same for each lexical item, regardless of which form always wins in the target language (indicated by the ☞)

65) *Output distributions for the exceptional coda rankings*

input	output	ranking (a)	ranking (b)	ranking (c)	ranking (d)
(i) pak	☞ pak	68.1%	65%	74.3%	71.7%
	pa	31.8%	35%	25.7%	28.2%
(ii) blag	blag	68.1%	65%	74.3%	71.7%
	☞ bla	31.8%	35%	25.7%	28.2%

6.2 Learning exceptions: GLA-related approaches

The previous section showed that a straightforward implementation of the GLA will treat both the variable and exceptional coda grammars the same – in both cases it will learn a variation grammar. I have illustrated this point primarily to add to the arguments against the GLA's lack of stored errors – because without stored errors that categorically do or don't follow a generalization, a language with exceptions looks just like a variable one.

There is at least one proposal in the GLA-related literature that is built especially to handle grammars with exceptions. This is the model of Zuraw (2000), which I will refer to here as the Full Listing model for reasons that will become clear.

In the Full Listing model, the GLA is used to learn the stochastic ranking of constraints that govern lexical subregularities.[19] After learning with the GLA, such constraints overlap and so produce variable results: but in Zuraw's system these constraints are ranked low enough that they can only have any effect on the choice of output for *words that do not have a lexical entry* – that is, new coinages, nonce words or unfamiliar words heard uttered by another speaker. In contrast, the shape of every input in the lexicon (whether deemed regular or exceptional by the linguist) is protected by high-ranking I-O lexical faithfulness constraints.[20]

In fact, this model exploits the GLA's ability to learn a variation grammar from categorically regular and exceptional inputs in a very clever way. What the GLA does for the Full Listing learner is tinker with the ranking of the constraints that prefer and

[19] C.f. the ordinal OT approach of Pater and Coetzee (2005), designed for a related kind of learning off the lexicon.

[20] As well as a constraint UseListed, which ensures that learners use their stored lexical forms as inputs, rather than creating new inputs for known words and letting low-ranked constraints affect their output.

disprefer particular processes, in tune with their lexical frequency, while steadily moving faithfulness up the hierarchy. Once IO-faith gets high enough, errors stop being made and learning is done. The resulting grammar is one in which, for everyday production purposes, IO-faith makes all the decisions. The learner has encoded all forms in the lexicon, both regulars and exceptions, and they will always be faithfully produced. But at the bottom of the hierarchy resides the variation grammar that the GLA learned. This ranking is the locus of speaker's knowledge about stastistical regularities in their lexicon, and it also perpetuates lexical tendencies across generations of learners by driving the frequency with which *novel* words enter the language as regular or exceptional.[21]

What this Full Listing model gives up, however, is the goal that the grammar be responsible for the phonological regularity or exceptionality of lexical items. The violation profile of each winner was used in learning to build the low-ranking stochastic grammar – but in the final adult grammar that profile is not important because IO-faith keeps forms identical to their stored forms. As a result, the Full Listing learner's grammar is not built to be restrictive, and does not rule out a Rich Base.

Hayes and Londe (2006)'s analysis of Hungarian stem-controlled vowel harmony adopts some key assumptions from the Full Listing model, but these authors also aim to build a restrictive grammar independent of the lexicon. They do so by proposing a two-step learner, splitting the learner's task into two consecutive parts: first, to acquire a restrictive phonotactic grammar of what is possible and impossible in Hungarian, and second, to acquire the statistical regularities of where the possible patterns of harmony occur, and how often. To accomplish the first task, they use a non-GLA learning algorithm: either BCD or Hayes' LFCD (Hayes, 2004; see my discussion in chapter 2

[21] I have done rather short shrift to this approach; see Zuraw (2000) chapter 2 for the whole story.

section 2), which shares with BCD the crucial properties of stored errors and the acquisition of ordinal rankings.

To accomplish their second learning task, the two-step learner uses the GLA, but with two large caveats. First, they use the constraint rankings acquired in phonotactic learning *as a priori rankings for GLA learning*; second, they remove faithfulness *entirely* from the constraint set that the GLA works with. Thus, all the GLA is allowed to learn is the relative frequency of various harmonic and disharmonic vowel sequences, via the frequency of their markedness violations.

The two-step Learner of Hayes and Londe also adopts the Full Listing assumption – namely that all derived lexical items are stored fully composed with their suffixes. In their analysis, each Hungarian stem is stored with the allomorph(s) that it is observed to take: the front suffix, the back suffix, or both in the case of stems that the authors call 'vacillators'. In this way the two-step learner's grammar is indeed responsible for deriving forms in the usual generative way. On the one hand, the ranking learned by BCD or LFCD in their first acquisition step ensures that stems stored with only one suffix will optimally keep that suffix in the output. On the other hand, vacillators are provided to the grammar as inputs with both suffix allomorphs, and since high-ranking IO-faith cannot choose between allomorphs[22] the low-ranking stochastic rankings learned by the GLA will choose between them (just as they do in Zuraw's approach to nonce words.)

Although Hayes and Londe do not discuss this brand of exceptionality: with the assumption of full allomorph listing and a generous notion of allomorphy, their model seems able to capture the exceptional coda grammar from section 6.1.3 above. However,

[22] See Kager (1999), Wolf (2005) on this assumption that inputs with two allomorphs map fully-faithfully to outputs with either allomorph.

it does so precisely because its set-up prevents the GLA from playing a role. In the first stage of phonotactic learning, the learner will determine that codas are possible and so acquire the ordinal ranking Faith >> NoCoda (just as in §6.1.3.) Upon discovering that words like [gri] come in fact from inputs like /grip/ (presumably from alternations), the learner must know to store *both* of these outputs as allomorphs of this root – thus, the learner's inputs will be as in (66) below:

66) *The exceptional coda language in a Hayes and Londe-like system*

a) the inputs	b) the outputs (in citation form)
/pak/	[pak]
/pa/	[pa]
/blag/	[blag]
/bla/	[bla]
/{**grip, gri**}/	[gri]
/tro/	[tro]

With this lexicon, regular inputs with codas will retain them (67a) and exceptional inputs with two allomorphs will lose them via low-ranking NoCoda (67b):

67) a) *Deriving regular coda preservation…*

/pak/	Max	NoCoda
(i) ☞ pak		*
(ii) pa	*!	

b)*… and exceptional coda deletion*

/{grip, gri}/	Max	NoCoda
(i) grip		*!
(ii) ☞ gri		

One outstanding question is the extent to which this approach's brand of allomorphy can handle all the attested types of lexical exceptions (for a sample of real cases see Pater, to appear.) What is clear from this example, though, is that the Two-Step learner of Hayes and Londe acquires this exceptional grammar but uses a re-ranking algorithm by using a BCD-like ranking, and not the GLA.

6.3 Learning variation without the GLA: a BCD approach

Wee have just seen that in contrast to the GLA, a Biased Constraint Demotion learner equipped with a Support can indeed learn a exceptionality grammar. As discussed in chapter 2 §2.2 Pater (to appear) uses Inconsistency Detection and Biased Constraint Demotion to successfully learn a grammar that encodes exceptionality using lexically-indexed constraints (see also Kager, in press; Flack, to appear.) In a related vein, Pater and Coetzee (to appear) also use a similar BCD approach to learn a grammar that encodes lexical subregularities; see also Becker, 2006.

On the other hand, the BCD learner has no extant mechanism for acquiring a variation grammar. Recall that the variable ESL learner I proposed in chapter 2 is variable only as long as it is learning – once all the necessary errors have been added to the final Support and the Error Cache is cleared for the final time, there are no way to cause variation because there is only one ranking. More generally, it is not necessarily clear what an end-state variation grammar should look like in an ordinal OT system, regardless of how it should be learned (on the end-state question, see Nagy and Reynolds, 1995; Inkelas, Orgun and Zoll, 1997; Anttila, 1997, 2002; Pater and Werle, 2003; Pater to appear.) Since the GLA is inherently good at learning such grammars, this chapter's general critique of the GLA model demands some discussion of how an ordinal OT grammar might be made to capture end-state variation – and also learn it.

The approach to variation in Error-Selective Learning in chapter 3§5 started from the claim that variation in an ordinal OT system must be derived by switching between multiple *grammars* – unlike a GLA-style grammar, which can generate different *rankings* to use in each run of EVAL from a single grammar. To re-use the example of the coda

variation grammar in §6.1.2 but now in ordinal OT terms: the speaker of this language will have to sometimes use a ranking in which NoCoda >> Max, and other times use a ranking in which Max >> NoCoda.

Along these lines, one well-known approach to variation in ordinal OT is to use partially-unranked constraints, as proposed by Anttila (1997, 2002); the version I will adopt here is the (2002) model. In this approach, grammars consist of sets of specified constraint rankings which are consistent with all of the language's variation, and with all remaining constraint orderings left unspecified. This means that the speaker of the variable coda language (using just these three constraints) has a grammar in which the only specified ranking is NoCoda >> Dep, and the ranking of Max is left unspecified. Every time the speaker uses the grammar, they will randomly pick a full ordering of the constraints, which means that Max will get ranked either above or below NoCoda:

68) *The variable coda grammar in the partially-unordered constraints model (adapted from Antilla 2002)*

a)	The grammar:	specified rankings:	NoCoda >> Dep
		unranked constraints:	{Max}
b)	The full orderings:		NoCoda >> Dep >> **Max**
			NoCoda >> **Max** >> Dep
			Max >> NoCoda >> Dep

(continued on the next page)

c) The variation these fully-ordered rankings create:

(i) first two rankings: (ii) third ranking:

/pak/	NoCoda	Max	Dep
(i) pak	*!		
(ii) ☞ pa		*	

/pak/	Max	NoCoda	Dep
(i) ☞ pak		*	
(ii) pa	*!		

There are some known difficulties with this model,[23] but does provide us with an explicit model of ordinal OT variation with which we can ask the relevant learning question: How could a BCD learner come to adopt the grammar in 68a)?

Pater (2005) provides a proposal for learning variation with the T/S algorithm which, like with everything else in this dissertation, proceeds by examining the properties of its Support. The suggestion in Pater (2005) for how the learner could conclude the need for the partially-unordered rankings of a variation grammar is to notice that the same input has created two or more winners – that is, *if it detects inconsistency among errors with identical inputs*. Such a Support is built below, adapted from the errors fed to the GLA back in table 60):

[23] As pointed out by Pater (to appear), one issue with using partially-unranked constraints to model exceptionality is that at one level it equates variation and exceptionality, just as the GLA does. What we have already seen that this makes the wrong predictions about a truly exceptional grammar – and Pater (to appear)'s more general argument is that exceptional and variable phonological patterns can easily exist one without the other and so require a grammar (and a learner) that knows the difference between them. In any event, I have presented only the bare bones of both the theory and the dissent here – see the Antilla and Pater references given above for a fuller view of the debate. See also Inkelas, Orgun and Zoll (1997) for a related but somewhat different view of partially-unranked constraints to handle variation, though not exceptions. For another issue, see footnote 25.

69) *Errors for the BCD learners of the variable coda grammar*

input	winner ~ loser	NoCoda	Max	Dep
(i) /pak/	pak ~ pa	L	W	e
(ii) /pak/	pa ~ pak	W	L	e
(iii) /blag/	blag ~ bla	L	W	e
(iv) blag/	bla ~ blag	W	L	e

(Note two important things about this Support: first, that the frequency of each variant and its associated error is being ignored in this discussion,[24] and second that this learner is at a state where it has already determined the correct underlying representations, including the unfaithful ones as in rows (ii) and (iv), just as was assumed in the previous GLA discussion.)

On the basis of a Support like (69), Pater (2005) proposes that the learner acquire two fully-specified rankings, each of which covers one of the two inconsistent ERC rows. Below, I spell out a slightly different way of learning from this inconsistency, relying on the partially-unranked constraints we have already used to capture variation in this section.

The BCD learner will fail to find a ranking for the Support in 69) as soon as it gets started: it cannot install a single constraint, because its Markedness constraint NoCoda prefers losers, and neither faithfulness constraint prefers only winners. To check whether its inability to find a grammar is due to variation in the data (instead of exceptionality, incorrect representational assumptions or anything else), the learner might now consider each set of errors in the Support *that have the same input,* and attempt to

[24] In the Antilla (2002) model used in 68), the frequency with which each variant is chosen is a function of how many rankings of the unordered constraints choose that variant. The tableaux in 68c) show that, given this constraint set and grammar, coda deletion should be chosen 66% of the time, coda faithfulness should be chosen 33% of the time, and no other frequency of variation is predicted. While more constraints could be introduced to prefer one variant more often than the other, the connection between variants and their frequency is clearly dealt with much more elegantly in the GLA system, where just two constraints can create many different patterns of proportional variation as a function of the distance between their ranking values.

find a ranking for just that set. In the Support in 69) above, this means grouping together the first two ERC rows, and the last two ERC rows – and right away we can see that both subsets will again be found inconsistent, because they each contain conflicting information about the ranking of NoCoda and Max:

70) *The Support built e.g. only for the input /pak/ is still inconsistent*

input	winner ~ loser	NoCoda	Max	Dep
(i) /pak/	pak ~ pa	L	W	e
(ii) /pak/	pa ~ pak	W	L	e

In the face of this inconsistency among errors for the same input, the learner can at least be sure that if these two input-winner pairs are right, then grammar it is learning contains variation. In the view I have adopted here, the effect of this discovery must be that the learner chooses a set of constraints to be unordered in the grammar.

Looking at 70), it is clear that the conflict between NoCoda and Max are responsible for the inconsistency. How will the learner see this? Recalling the Antilla (2002) model of variation, the method I propose is that the leaerner first builds rankings for each individual ERC row in the single-input Support (like 70), and then builds a grammar which specifies all and only the constraint orderings common to all these ERC-specific rankings.

If we take this first step for the two errors in 70), we will nearly get the two ERC-specific rankings below:

71) *Building rankings for each ERC row variant*

a) the first ERC from 70): which builds this grammar:

Input	winner ~ loser	NoCoda	Max	Dep
(i) /pak/	pak ~ pa	L	W	e

Max >> NoCoda >> Dep

b) the second ERC from 70): which builds this grammar:

Input	winner ~ loser	NoCoda	Max	Dep
(ii) /pak/	pa ~ pak	W	L	e

NoCoda >> Max, Dep

To take the second step, the learner can compare each constraint pair and only specify those whose ordering is the same in all the ERC-specific ranking. As spelled out in 72) below, this will build a grammar like in 68), with just NoCoda >> Dep specified:

72) *Building a variation grammar from the rankings in 71)*

a) NoCoda and Dep: NoCoda >> Dep in 71a)
NoCoda >> Dep in 71b) ... specified in the grammar

b) NoCoda and Max: Max >> NoCoda in 71a)
NoCoda >> Max in 71b) … so left unspecified

c) Max and Dep: Max >> Dep in 71a)
Max, Dep in 71b)[25] … so left unspecified

d) Final grammar: NoCoda >> Dep
{Max}

Admittedly: this procedure for collapsing ERC-specific rankings into a Antilla (2002) variation grammar works fine for a system with only 3 constraints, but would no doubt require refinement when scaling up to learning grammars with more constraints

[25] In this comparison, I have chosen to treat the constraints left unranked within a stratum by the BCD algorithm as different from those crucially ranked. But note that the BCD algorithm puts Max and Dep in the same stratum here *not* because it is crucial that they be unranked – recall Step 4 of the algorithm in chapter 2, which simply dumps all remaining IO-faithfulness constraints at the bottom of the hierarchy in one stratum.) If we instead chose to adopt the specified ranking in this situation – here, Max >> Dep – we would make the same predictions for the kinds of variation the language shows (coda deletion vs. faithfulness), but different predictions about the frequency of each variant (see previous footnote.) I leave this interesting difference for further work.

(and possibly more complicated interactions.) Nevertheless we have at least seen that an explicit procedure can be defined for the BCD learner to build a partially-ordered variation grammar.

Assuming that this procedure or something like it will allow the learner to get from a Support like 69) to the grammar in 72d), there must now also be some revision to the normal workings of the learner. In all subsequent attempts to feed observed forms through the current grammar, the learner now has a *number* of multiple rankings that all instantiate that current grammar. It is no longer clear how the learner should proceed in parsing observed outputs, and determining when an error has been generated.

Pater (2005) suggests that error generation in a variation grammar could proceed by assuming that if an observed output can be produced faithfully by at least *one* of the rankings consistent with the current grammar, it does not count as an error. However, this technique will be insufficient because it will prevent the learner from seeing any lexical exceptions once a variation grammar has been adopted. Just because /pak/ and /blag/ retain their codas in a variable fashion does not mean that every coda in the language behaves this way; the partially-ordered grammar will always have *one* ranking that does each, but it will never notice on its own that e.g. /blak/ always retains its coda while e.g. /pag/ never does (see again Pater to appear.)

At this point, the best way to learn using a variation grammar and the Error-Selective BCD learner must be left as an open issue. But given what has come before, I suggest that a likely approach to noticing the difference between exception and variable inputs will again come from the BCD learner's analysis of the Support and its associated rankings, as well as the range of ERC rows that share common inputs.

Overall, this section's discussion has demonstrated that an enriched analysis of the Support can at least provide the BCD learner with the knowledge that phonological variation is part of its target ordinal OT grammar. The method by which this learner moves from an inconsistent set of ERCs with a common input to the right variation grammar, and how adopting a variation grammar at an intermediate stage affects the further course of BCD learning, remain open questions.

6.4 Summary

This section has compared the GLA and BCD approaches to the discovery of exceptionality and variation in target grammars. With respect to these two areas, both algorithms have their strong and weak points. On the one hand, BCD learning is better suited to diagnose and encode categorical exceptions, because unlike the GLA it notices that the conflicting ranking information it is receiving comes from different inputs. On the other hand, GLA learning (of numerical OT grammars) is better suited to capturing variation because unlike an ordinal OT learner it can easily respond to conflicting evidence by gradually overlapping the ranking of relevant constraints.

7. Chapter Summary

This chapter has discussed the Gradual Learning Algorithm, an alternative approach to the OT learning questions treated in chapters one and two. While the GLA is inherently a learner that goes through intermediate stages, I have argued that it is not the best way to model phonotactic learning. I have demonstrated its restrictiveness problems with specific faithfulness relations, arguing that they cannot be fully treated using ranking

biases, and also pointed to its inability to produce some Specific-F intermediate stages. I have also argued that the choice not to retain its errors, in contrast to BCD's storage of errors in the Support, makes the GLA unable to recover from early winner misparses and the superset rankings they cause.

Finally, I have discussed the extent to which the GLA and the BCD learners can each handle grammars with exceptional forms or variation within forms. To return to one of the broader themes of this dissertation: this discussion has served to highlight that whether our eventual grammar uses ordinal or numerical rankings, it is still crucial that the learner keep a memory of its errors to determine the true nature of the pattern it is attempting to learn.