The Development of the Germanic Preterite System:
Learnability and the Modeling of Diachronic Morphophonological Change

Ryan Sandell & Sam Zukoff
University of California, Los Angeles & Massachusetts Institute of Technology
ryan.sandell@gmail.com, szukoff@mit.edu

5 January 2017

1. The Problem of Germanic Strong Verbs

1.1. **Driving Question**: Why and how could learners fail to acquire **reduplication** as a marker of a morphosyntactic category when it was present in the learning data?

1.1.1. In Proto-Indo-European (PIE), **past tense** is characterized by prefixing reduplication.

1.1.2. In (Proto-)Germanic, **past tense** is characterized by a variety of stem alternations.

1.1.3. These two systems undoubtedly have a direct historical connection, so what caused the system to change in the way it did?

1.2. Present-day Germanic languages display "irregular" verbs that form their past tense (henceforth PRETERITE) forms by stem alternations, as in the English examples in (1).

(1) "Irregular" Verbs in Present-Day English

a. Eng. PRESENT *sing* : PAST *sang*
b. Eng. PRESENT *give* : PAST *gave*
c. Eng. PRESENT *bite* : PAST *bit*
d. Eng. PRESENT *hold* : PAST *held*

- The unproductive patterns of "irregular" verbs contrast with the productive formation of past tense forms with a coronal stop suffix: English /-d/, German /-tə/, etc. Cf. Pinker and Prince 1992, Albright and Hayes 2003.

1.3. These stem alternations can be reconstructed to Proto-Germanic (PGmc.), and remained productive in the oldest attested Germanic language, Gothic:

(2) Gothic Preterite System

	Root Types		PRESENT	PRET. SG.	PRET. PL.	Gloss
Non-reduplicated stems	CeRC roots	(Class I–III)	*beit-an*	*bait*	*bit-un*	'bite'
	CeC roots	(Class IV–V)	*gib-an*	*gaf*	*geːb-un*	'give'
	CaC roots	(Class VI)	*far-an*	*foːr*	*foːr-un*	'travel'
"Reduplicated" stems	CaCC roots	(Class VIIa)	*hald-an*	*hehald*	*hehald-un*	'hold'
	CV:C roots	(Class VIIb)	*leːt-an*	*leloːt*	*leloːt-un*	'let'

1.3.1. The Gothic preterites represent a system of **phonologically conditioned stem modification** for the purpose of obtaining *phonological* contrast between *morphologically* distinct stems.

1.3.2. These modifications are primarily vowel alternations, but, under certain circumstances, instead involve reduplication.

1.4. This system of preterite marking in Gothic/PGmc. differs substantially from that of its formal and functional ancestor, the PIE "PERFECT".

(3) The PIE perfect (as represented by Sanskrit)

	Proto-Indo-European (> Sanskrit)		Gothic	
Example Root	*/bʰeid/	(> Skt. /bʰəjd/ 'split')	/bejt/ 'bite'	
Preterite Sg. stem	*<u>be</u>-bʰóid-e	(> Skt. <u>bi</u>-bʰéːd-a)	ˈbait,	not **ˈ<u>be</u>-bait
Preterite Pl. stem	*<u>be</u>-bʰid-ŕ	(> Skt. <u>bi</u>-bʰid-úr)	ˈbit-un,	not **ˈ<u>be</u>-bit-un

1.4.1. The PIE perfect was obligatorily marked by **reduplication**.

1.4.2. It also displayed accentually-conditioned vowel alternations ("ablaut"), but these did not directly bear morphological significance.

1.5. **Question:** How did the system of reduplication found in PIE develop into the system attested in Gothic?

1.6. **Proposal:** A change in the Proto-Germanic accentual system rendered the original reduplicative underlying representation (/RED$_{PRET}$/) irrecoverable for certain preterite forms, leading learners to restructure the PRETERITE derivation wholesale:

1.6.1. The preterite morpheme has a phonologically null UR, /Ø$_{PRET}$/ (except possibly in Class VII)

1.6.2. Stem modifications are triggered by constraints on stem contrast: REALIZE MORPHEME (RM)

1.6.3. RM constraints interact with markedness and faithfulness to determine stem modification for roots of various phonological shapes

1.7. We model this change using Maximum Entropy learning simulations:

1.7.1. Multi-generation agent-based learning using Robust Interpretive Parsing succeeds in generating the Gothic system from the PIE input data, modulo the Proto-Germanic accent shift.

1.7.2. The assumptions necessary for the model provide specific evidence for the way various types of **bias** should be incorporated into phonological and morphological learning.

2. The Gothic System

2.1. Gothic **strong verbs** are divided into seven classes, distinguishable by phonological properties of the root. These properties in turn determine how the preterite is formed. The seven classes are shown in (4).[1]

(4) (Pre-)Gothic strong verb system

Root Shape	Class	1SG.PRES	3SG.PRET	1PL.PRET	GLOSS
/CeRC/	I	bejt-a	bajt	bit-um	'bite'
	II	kews-a	kaws	kus-um	'choose'
	III	bend-a	band	bund-um	'bind'
/CeC/	IV	nem-a	nam	neːm-um	'take'
	V	geb-a	gaf	geːb-um	'give'
/CaC/	VI	dab-a	doːf	doːb-um	'happen'
/CV{ː,C}C/	VIIa	hajt-a	he-hajt	he-hajt-um	'call'
	VIIb	sleːp-a	se-sleːp	se-sleːp-um	'sleep'
	VIIc	floːk-a	fe-floːk	fe-floːk-um	'bewail'
	VIId	leːt-a	le-loːt	le-loːt-um	'let'

[1] We make one simplifying move: Gothic underwent a merger of */e,i/ > /i/ in its late pre-history. This opacifies some of the interactions relevant to our discussion. Therefore, we operate with the representations prior to this sound change, i.e. "Pre-Gothic".

2.1.1. Classes I–V, which have roots in /e/, make their preterite singulars in [a].

2.1.2. Classes I–III, which are CeRC roots, make their preterite plurals by deleting the root vowel and vocalizing the root-medial sonorant.

- Class I = roots with underlying medial /j/: [bitum] ← //bjt-um//
- Class II = roots with underlying medial /w/: [kusum] ← //kws-um//
- Class III = roots with underlying medial non-glide sonorant /m,n,r,l/: [bundum] ← //bnd-um//

2.1.3. Classes IV–VI, which are all CVC roots, make their preterite plurals by lengthening the root vowel; Class VI also does this for its preterite singular form.

2.1.4. Class VII forms both preterite singular and plural through reduplication.

2.2. **Generalizations:**

(i) Preterite stems are *always* distinct from their present stems.
(ii) Preterite plural stems are *usually* distinct from their preterite singular stems.

2.3. This type of contrast can be effected, even in the absence of segmental material belonging to some underlying affix, using Realize Morpheme constraints (based on Kurisu 2001). Since the two generalizations differ in their applicability, we must employ distinct constraints, as in (5):

(5) Realize Morpheme constraints

a. **Realize Morpheme: preterite (RM:pret)**
Assign a violation mark * for any preterite stem which is not phonologically distinct from the present stem formed from the same root.

b. **Realize Morpheme: plural (RM:pl)**
Assign a violation mark * for any preterite plural stem which is not phonologically distinct from the preterite singular stem formed from the same root.

2.4. We posit that the strong preterite has an underlyingly null phonological exponent. In such a case, the Realize Morpheme constraints can only be satisfied through an unfaithful mapping.

2.4.1. The selection of the particular unfaithful mapping for a particular form depends on the phonological characteristics of the root, and the constraint ranking.

2.4.2. We now briefly illustrate this for several of the core cases. (See Zukoff and Sandell 2015 for full analysis.)

****

2.5. Class I–V preterite singulars

2.5.1. The Class I–V preterite singulars represent the basic case. They employ *vowel backing* (/e/ → [a]) as their means of satisfying RM:pret.

2.5.2. To model this and other similar changes, we use DepFeature constraints referencing specified feature values (cf. Casali 1996, *a.o.*).

(6) Faithfulness constraints

a. **Dep[+back]-IO:** One * for each [+back] feature in the output which was not present in the input.

b. **Dep[+high]-IO:** One * for each [+high] feature in the output which was not present in the input.

c. **MaxV-IO:** One * for each vowel in the input which lacks a correspondent in the output.

(7) PRET.SG. of Class II (also Class I–V)

/kews, PRET/ ; BASE: PRES [kews-]		RM:PRET	DEP[+high]-IO	MAXV-IO	DEP[+back]-IO
a.	kews	*!			
b.	kiws		*!		
c.	kus (← //kws//)			*!	
d. ☞	kaws				*

2.6. Class I–III preterite plurals

2.6.1. In the preterite plural, both RM constraints will be active.

2.6.2. The activity of RM:PL rules out vowel backing, since it has already been claimed by the singular.

2.6.3. Therefore, the preterite plural must settle for the next best option, which is *vowel deletion*:

(8) PRET.PL. of Class II (also Class I–III)

/kews, PRET, um/ ; BASES: PRES [kews-], PRET.SG: [kaws-]		RM:PRET	DEP[+high]-IO	RM:PL	MAXV-IO	DEP[+back]-IO
a.	kewsum	*!				
b.	kiwsum		*!			
c. ☞	kusum (← //kwsum//)				*	
d.	kawsum			*!		*

2.7. Class IV–V preterite plurals

2.7.1. In Classes IV & V, the preterite plural is formed not by vowel deletion, but rather by *vowel lengthening*.

2.7.2. This repair occurs, contrary to the normal preference for vowel deletion, in order to avoid creating the consonant cluster that would result from vowel deletion in roots of the shape /CeC-/.

(9) a. **DEP[+long]-IO:** One * for each output [+long] feature which was not present in the input.

b. ***CC:** One * for each consonant cluster in the output.

(10) PRET.PL. of Class V (also Class IV)

/geb, PRET, um/ ; BASES: PRES [geb-], PRET.SG: [gab-]		RM:PRET	*CC	RM:PL	DEP[+long]-IO	MAXV-IO	DEP[+back]-IO
a.	gebum	*!					
b.	gbum		*!			*	
c.	gabum			*!			*
d. ☞	ge:bum				*		

2.8. Class VII preterites: reduplication

2.8.1. Class VII roots form their preterites through *reduplication*.

2.8.2. Class VII includes roots with underlying long vowels (Class VIIb-d), and also roots with underlying root /a/ followed by two consonants (Class VIIa).

- Underlying /a/'s and long vowels cannot be deleted (high-ranking MAX[+back]/DEP[-back] and MAX[+long]/ DEP[-long], respectively).
- If such roots were to undergo lengthening, the result would be a superheavy syllable.

2.8.3. Given a sufficiently high ranking of *SUPERHEAVY, lengthening will be a non-optimal repair in precisely these cases. The constraint violated instead is INTEGRITY-IO.

(11) a. **INTEGRITY-IO:** One * for each input segment which has multiple correspondents in the output.
b. ***SUPERHEAVY:** One * for each superheavy (i.e. trimoraic) syllable.

(12) PRET.SG. of Class VII

/hajt, PRET/ ; BASE: PRES [hajt-]		RM:PRET	*SUPERHEAVY	MAX[+back]	INTEGRITY-IO	RM:PL	DEP[+long]	MAXV	DEP[+back]-IO	*[a]	IDENT[±back]-BR
a. hajt		*!									
b. ho:jt (← //ha:jt//)			*!				*				
c. hit (← //hjt//)				*!				*			
d. hejt				*!							
e.	☞ hehajt				**						*
f. hahajt					**					*!	

3. From Proto-Indo-European to Proto-Germanic

3.1. Based primarily on the evidence from Sanskrit and Greek, as shown in (13), the PIE "perfect" (> Germanic "preterite") is reconstructed with two properties:

3.1.1. Obligatory reduplication (prefixal CV), which is the morphological exponent of the morphosyntactic feature PERFECT.

3.1.2. Root-vowel alternations conditioned by the placement of the accent (cf. Kiparsky 2010, Forthcoming):
- **Singular**: Root /e/ → [ó] (root is accented by pre-accenting suffixes, suffix is unaccented)
- **Plural**: Root /e/ → Ø (lexically-accented suffix, root is unaccented)

(13) The PIE perfect

	PIE	Sanskrit	Greek
Example root	*/bʱejt-/	/bʱəjt-/ 'split'	/leip-/ 'remain'
Perfect Sg. stem	*<u>be</u>-bʱóid-e	<u>bi</u>-bʱéːd-ə	<u>le</u>-lóip-e
Perfect Pl. stem	*<u>be</u>-bʱid-ŕ	<u>bi</u>-bʱid-úr	<u>lé</u>-lip-on

3.2. We argue that the long vowels in the Gothic Class IV & V preterite plurals – *ˈneːm-un*, *ˈgeːb-un*, etc. – result from reduplicated forms which were, in PIE, subject to two phonologically-driven deletion processes:[2]

(14) i. *Zero-grade ablaut*: deletion of pre-tonic root vowels
- Occurs only in plural, where suffixes are accented.

ii. *Repeated-consonant deletion + compensatory lengthening*: $C_\alpha VC_\alpha C \rightarrow C_\alpha V{:}C$
- On the constraint triggering the deletion process, see Zukoff (2014, 2015, forthcoming, in progress), Sandell 2015; cf. Schumacher 2005. This constraint will be involved in the learning models below.

[2]These forms have direct comparanda in Sanskrit, e.g. *neːm-úr*, *seːd-úr*. The Sanskrit and Gothic forms cannot directly continue PIE *[neːmúr], *[seːdúr], etc., as these would yield Sanskrit *[naːmúr], *[saːdúr], etc. Rather, these must be built synchronically as such in the separate languages, having similar constraints and rankings.

3.3. Reduplication and vowel deletion feed the consonant-deletion process,[3] as shown in (15):

(15) PIE /RED, nem, ṕ/ $\xrightarrow[(copy\,CV)]{\text{reduplication}}$ *ne-nem-ṕ* $\xrightarrow[(e \rightarrow \emptyset / \_C_o\acute{V})]{\text{zero-grade ablaut}}$ *nenmṕ* $\xrightarrow[(VC_\alpha \rightarrow V{:}/C_\alpha\_C)]{\text{C-deletion + CL}}$ [ne:mḿ]

3.4. Between PIE and Proto-Germanic, there was a complete remodeling of the accentual system, as in (16): mobile, lexically-determined accentuation is replaced with phonologically-fixed initial stress (cf. Halle 1997).

(16) PIE Mobile Pitch Accent ⇒ Proto-Germanic Fixed Stress Accent

a.	PIE *[pəĥté:r]	>	PGmc. *[ˈfaðer]	'father (NOMINATIVE)'
b.	PIE *[pəĥtr-ós]	>	PGmc. *[ˈfaðraz]	'father (GENITIVE)'
c.	PIE *[pó:t-s]	>>	PGmc. *[ˈfo:tiz]	'foot (NOMINATIVE)'
d.	PIE *[ped-ós]	>>	PGmc. *[ˈfo:tijaz]	'foot (GENITIVE)'
e.	PIE *[hés-ti]	>	PGMc. *[ˈesti]	'is'
f.	PIE *[hs-énti]	>	PGmc. *[ˈsinti]	'are'

3.5. This change in accentuation eliminates the conditioning environment for the vowel deletion rule: the step from intermediate *ˈne-nem-un* to intermediate *ˈne-nm-un* is rendered opaque, as seen in (17).

(17) PGmc. /RED, nem, un/ $\xrightarrow{\text{redup. + stress}}$ [*ˈne-nem-un* $\xrightarrow[???]{\text{zero-grade ablaut}}$ *ˈnenmun*] $\xrightarrow{\text{C-deletion + CL}}$ [ˈne:mun]

3.6. At the point of the Germanic stress shift, learners are thus faced with an alternation between inherited 3SG.PRET *[ˈne-nam] (< *[ne-nóm-e]) : 3PL.PRET [ˈne:m-un] (<< *[ne:m-úr]), yet they are (by hypothesis) unable to connect the reduplication in the singular with the apparent non-reduplication in the plural.

3.7. The singular form which we observe in Gothic is simply [ˈnam] w/o reduplication, not *[ˈne-nam]. What accounts for this change? **Why is reduplication lost** on this form, and generally throughout the system?

3.8. We will now demonstrate, using computational learning simulations, that the inability to recover reduplication in cases like [ˈne:m-un] plays a role in triggering the loss of reduplication throughout the system.

4. Learnability Problems: How does PIE become Gothic?

4.1. Our learning objective: Find conditions under which the *observed* winners are *not learned*, and instead replaced by new output forms under a (possibly) new grammar.

- 4.1.1. Specifically: some generation of Proto-Germanic learners received preterite inputs exhibiting both **reduplication** and **ablaut** like [ˈnenam] 'he took', but *failed to acquire a grammar that could generate this observed datum*.
- 4.1.2. Instead, these learners selected a form like [ˈnam], without reduplication.
- 4.1.3. **Question:** What learning conditions will lead learners to choose an output different (like [ˈnam]) than their observed data (*[ˈnenam])?
- 4.1.4. More concretely: when presented with [ˈnenam] as a learning datum, can the learner arrive at a grammar /nem, $\varnothing_{\text{PRET}}$/ → [ˈnam] (as in Gothic)?

[3] It is unclear if it is possible to fully analyze this pattern in a non-serial OT derivation.

(18) Creating Germanic Strong Verbs (compare Table 4 above)

		Pre-Proto-Germanic		Gothic		
Root Shape	**Class**	3SG.PRET	3PL.PRET	3SG.PRET	3PL.PRET	Changes
/CeRC/	I	*[be-bájt]	*[be-bit-ún]	['bajt]	['bit-un]	Loss of /RED/
	II	*[ke-káws]	*[ke-kaws-ún]	['kaws]	['kus-un]	
	III	*[be-bánd]	*[be-bund-ún]	['band]	['bund-un]	
/CeC/	IV	*[ne-nám]	*[ne:m-ún]	['nam]	['ne:m-un]	Loss of /RED/ in Sg.
	V	*[ge-gáb]	*[ge:b-ún]	['gab]	['ge:b-un]	
/CaC/	VI	*[de-dáb]	*[de:b-ún]	['da:b]	['da:b-un]	Loss of /RED/; [a:] in Pl.
/CV{:,C}C/	VII	*[he-hájt]	*[he-hit-ún]	['he-hajt]	['he-hajt-un]	Loss of Ablaut in Pl.

4.2. **Our Method:** Pair the constraint grammar developed for Gothic in Section 2 with the reconstructed surface forms of Pre-Proto-Germanic seen in Table 18, with the objective of generating the Gothic forms.

4.2.1. Between Pre-Proto-Germanic and Gothic, the following developments are predicted:

- the singular stem of Classes I–VI loses reduplication.
- the plural stem of Classes IV and V remains the same.
- the plural stem of Class VI changes to show [a:] rather than [e:].
- the plural stem of Class VII no longer shows vowel deletion.

4.2.2. After initial training on data resembling Pre-Proto-Germanic (Pre-PGmc.) forms, learners will move towards building grammars that resemble the constraint ranking and winning candidates of Gothic.

- The crucial difference between Pre-PGmc. and PGmc./Gothic is that ablaut in Pre-PGmc. is driven by markedness constraints.
- A grammar including accentually-conditioned markedness constraints driving ablaut is stable.
- Conversely, where alternations in vocalism cannot be attributed to surface-true phonotactic distributions, the phonological grammar is unstable.

4.2.3. Without independent markedness constraints to drive ablaut, adult winners like ['bebajt], showing both reduplication and ablaut, cannot win. See Tableau (19).

(19) Other Output ⇒ Input Mappings Harmonically Bound Pre-PGmc. Preterites

|bejt, PRET, SG|	RM:PRET	INTEGRITY-IO	DEP[+back]-IO	IDENT[back]-BR	*[a]
a. ☞	['bebajt] ⇒ /RED$_{PRET}$, bejt/		2	1	1	1
b.	['bajt] ⇒ /bejt, ∅$_{PRET}$/			1		1
c.	['bebejt] ⇒ /RED$_{PRET}$, bejt/		2			
d.	['bejt] ⇒ /bejt, ∅$_{PRET}$/	1!				

- The desired adult winner, Candidate (a), has a superset of the violations of Candidates (b) and (c).
- Therefore, there exists no grammar under which Candidate (a) can always win (i.e., receive 100% of the probability mass).

4.3. **The Learning Model**

4.3.1. All learning simulations conducted in Praat (v. 5.4) (Boersma and Weenink 1992–2014).

4.3.2. The optimal set of {Surface Winner, Underlying Representation, Constraint Ranking} is deduced from surface forms using ROBUST INTERPRETIVE PARSING (Tesar and Smolensky 2000: Ch. 6, Jarosz 2013).

- When presented with a SR, the learner identifies the most harmonic SR ⇒ UR mapping under the current grammar for that SR.

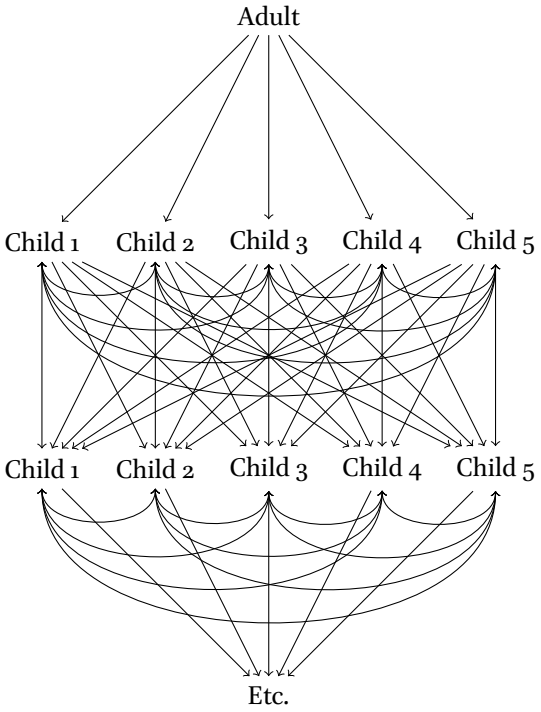
- The learner demotes the ranking constraints of constraints violated by this most harmonic SR ⇒ UR mapping, and promotes the ranking of constraints violated by the current winning SR ⇒ UR mapping in the tableaux, if different from the learning datum.

4.3.3. Candidate probabilities are evaluated, and the overall grammar is optimized, according to Maximum Entropy (Goldwater and Johnson 2003, Jäger 2007, Hayes and Wilson 2008). Constraint weights are updated using the Gradual Learning Algorithm Boersma and Hayes 2001, Boersma and Pater 2013).

4.3.4. Learning is iterated and agent-based (examples in phonology: Johnson 2011, Pater et al 2014).

- An initial batch of "children" (agents) is partly trained on "adult" seed data.
- "Children" (agents) of a generation train each other and provide outputs to train successive generations.

4.4. The Agent-Based Model



4.5. Tableaux, Constraints, and Training Data

4.5.1. The inputs to tableaux are bundles of semantic and morphosyntactic features, e.g., |bejt, PRET, PL|.

4.5.2. Candidates in tableaux are SR to UR mappings. Using ROBUST INTERPRETIVE PARSING, tokens of training data are matched to candidates fitting that winner.

4.5.3. Constraint violations are entered numerically.

(20) Sample Tableau for |bejt, PRET, SG|

	|bejt, PRET, SG|	RM:PRET	INTEGRITY	DEP[+back]-IO	IDENT[back]-BR	*[a]
a.		['bebajt] ⇒ /RED, bejt/		2	1	1	1
b.	☞	['bajt] ⇒ /bejt, ∅$_{PRET}$/			1		1
c.		['bebejt] ⇒ /RED, bejt/		2			
d.		['bejt] ⇒ /bejt, ∅$_{PRET}$/	1!				

4.5.4. The complete set of inputs is listed in (21).

(21) Learning Tableaux Inputs

Input	**Adult Form**	**Forms Represented**	Frequency in Seed Data
|bejt, PRES|	[bejt]	Present Stem for Classes I–V	278
|dab, PRES|	[dab]	Present Stem for Classes VI and VII	56
|bejt, PRET, SG|	[bebajt]	Preterite Singular Stem for Classes I–III	78
|bejt, PRET, PL|	[bebit]	Preterite Singular Stem for Classes I–III	78
|ber, PRET, SG|	[bebar]	Preterite Singular Stem for Classes IV–V	33
|ber, PRET, PL|	[be:r]	Preterite Plural Stem for Classes IV–V	33
|dab, PRET, SG|	[dedab]	Preterite Singular Stem for Class VI	22
|dab, PRET, PL|	[de:b]	Preterite Plural Stem for Class VI	22
|hajt, PRET, SG|	[hehajt]	Preterite Singular Stem for Class VII	34
|hajt, PRET, PL|	[hehit]	Preterite Plural Stem for Class VII	34

- Type frequencies used reflect the type frequency of that class of forms in the Gothic corpus.

4.5.5. Constraints are exactly those constraints employed in the analysis of Gothic in Section 2, with two additions penalizing repeated sequences:

(22) OCP′: Assign one violation * for each C_iVC_iC sequence. (see Section 3.2 above)

→ Needed to derive Class IV, V, and VI plural stems from URs with reduplication.

(23) *$[(CV)_i(CV)_i]$: Assign one violation * for each occurrence of identical CV sequences.

→ Needed to penalize forms like ['be-bejt] ⇒ /RED, bejt/ and ['ha-hajt] ⇒ /RED, hajt/.

- Learnable as an "unnatural" constraint (Hayes et al 2009), induced from the data; though may be related to more typical similarity avoidance constraints.

4.5.6. General Simulation Settings in Praat

- "Set decision strategy" as `ExponentialMaximumEntropy`
- When learning, the `Symmetric all` update rule is used.
- The following settings used in learning with ROBUST INTERPRETIVE PARSING (cf. Pater 2009):
  - `Evaulation noise`: 0
  - `Initial plasticity`: .1
  - `Plasticity decrement`: 1
  - `Number of plasticities`: 1
  - `Rel. placticity spreading`: 0
  - `Number of chews`: 1

4.5.7. The complete set of files used in the learning simulations is available online at: `https://github.com/rpsandell/LSA_2016_SandellZukoff`.

4.6. **The Initial State**

4.6.1. These simulations assume that the learner has already carried out the learning of phonotactic markedness constraints.

4.6.2. **Question:** What should the initial weights ("ranking value") and learning rates ("plasticity") be for different constraint types?

4.6.3. Standard assumption from learnability literature: Markedness ≫ Faithfulness.

4.6.4. Often implemented by putting all Faithfulness constraints in the lowest-ranked stratum or setting their weight to 0 (cf. Hayes 2004, Jesney and Tessier 2011, etc.).

- Intuitively, beyond the earliest stages of phonological acquisition, this is an oversimplification.
- We want to exclude [ba] (or the null parse) as the winner for all tableaux (which is what may result if Faithfulness plays no role in evaluating a candidate).
- Rather, Faithfulness violations can never be free – any divergence from an assumed UR must be penalized to some extent.
- For the same reason, we assume that Faithfulness constraints should not have a weight of 0 or a negative weight.

4.6.5. These simulations employ three types of constraints:

- Markedness (e.g., *SUPERHEAVY): set with initial Weight of 200, Plasticity of 1.
- Faithfulness (e.g., INTEGRITY): set with initial Weight of 100, Plasticity of 0.1.
- Morphological (e.g., REALIZE MORPHEME): set with initial Weight of 0, Plasticity of 10.

4.7. One deviation from the Gothic grammar is applied: RM:PLURAL is excluded.

- We exclude RM:PLURAL because the assessment of its violations depends upon the current winner for the corresponding singular form.
- For technical reasons, we cannot update constraint violation profiles on-line throughout the simulation; therefore, candidates that are ruled out in Gothic by RM:PLURAL are simply excluded from the learning tableaux in our simulations.

5. Walk-through of Simulation: Generation 1 Seed Data Training

5.1. **Class I–V Singulars:** Seed Training Datum: ['bebajt]

(24) Tableau for Class I–V Singulars

		|bejt, PRET, SG|	RM:PRET	*[(CV)i(CV)i]	INTEGRITY-IO	DEP[+back]-IO	IDENT[back]-BR	DEP[+long]-IO	*SUPERHEAVY	*[a]	MAXV-IO
a.	☞	['bebajt] ⇒ /RED$_{PRET}$, bejt/			2	1	1			1	
b.	>>	[bajt] ⇒ /bejt, ∅$_{PRET}$/				1				1	
c.		['bebejt] ⇒ /RED$_{PRET}$, bejt/		1	2						
d.		['bejt] ⇒ /RED$_{PRET}$ bejt/	1								1
e.		['bejt] ⇒ /bejt, ∅$_{PRET}$/	1								
f.		['babajt] ⇒ /RED$_{PRET}$, bejt/		1	2	2				2	
g.		['be:jt] ⇒ /bejt, ∅$_{PRET}$/			2			1	1		

5.1.1. Initial winner: ['bejt] ⇒ /bejt, ∅$_{PRET}$/

5.1.2. Training datum ['bebajt] maps best to ['bebajt] ⇒ /RED$_{PRET}$, bejt/.

- Results in demotion of: INTEGRITY, DEP[+back]-IO, IDENT[back]-BR, *[a]
- Results in promotion of: RM:PRET

5.1.3. When the weights of INTEGRITY and DEP[+back]-IO are sufficiently low, also results in promotion of *[(CV)$_i$(CV)$_i$].

5.2. Example Training Pathway for Class I–V Singulars

- Procedure: a model in the initial state is trained on sequential batches of 1000 tokens drawn from the types of adult tokens.

(25) Initial State

		bejt, PRET, SG		*Superheavy	*[a]	Integrity-IO	Dep[+back]-IO	Ident[back]-BR	Dep[+long]-IO	MaxV-IO	RM:pret	*[(CV)$_i$(CV)$_i$]	$\mathscr{H}$
		Ranking Value	200	200	100	100	100	100	100	0	0	
a.	['bebajt] ⇒ /RED$_{PRET}$, bejt/		1	2	1	1			1		-7.226×10^{86}
b.	[bajt] ⇒ /bejt, ∅$_{PRET}$/		1		1				1		-7.226×10^{86}
c.	['bebejt] ⇒ /RED$_{PRET}$, bejt/			2						1	-5.376×10^{43}
d.	['bejt] ⇒ /RED$_{PRET}$, bejt/							1	1		-5.376×10^{43}
e.	☞ ['bejt] ⇒ /bejt, ∅$_{PRET}$/								1		-1
f.	['babajt] ⇒ /RED$_{PRET}$, bejt/		1	2	2					1	-1.445×10^{87}
g.	['be:jt] ⇒ /bejt, ∅$_{PRET}$/	1		2			1				-7.226×10^{86}

- Without penalties for violation of RM:PRET, (e) is the most harmonic candidate.
- Violation of MaxV-IO by candidate (d) indicates deletion of a vowel timing unit in the reduplicative morpheme.

(26) After 1000 Tokens of Seed Data

		bejt, PRET, SG		*Superheavy	*[a]	RM:pret	*[(CV)$_i$(CV)$_i$]	Dep[+long]-IO	MaxV-IO	Dep[+back]-IO	Ident[back]-BR	Integrity-IO	$\mathscr{H}$
		Ranking Value	200	169.8	121.1	120.8	99.6	99.3	98.3	98.1	97.9	
a.	['bebajt] ⇒ /RED$_{PRET}$, bejt/		1						1	1	2	-3.952×10^{73}
b.	[bajt] ⇒ /bejt, ∅$_{PRET}$/		1						1			-3.952×10^{73}
c.	☞ ['bebejt] ⇒ /RED$_{PRET}$, bejt/				1						2	-3.901×10^{53}
d.	['bejt] ⇒ /RED$_{PRET}$, bejt/			1								-8.704×10^{53}
e.	['bejt] ⇒ /bejt, ∅$_{PRET}$/			1								-8.704×10^{53}
f.	['babajt] ⇒ /RED$_{PRET}$, bejt/		1		1				2		2	-7.904×10^{73}
g.	['be:jt] ⇒ /bejt, ∅$_{PRET}$/	1				1						-7.226×10^{86}

- Since *[a] ≫ *[(CV)$_i$(CV)$_i$], candidate (c) is more harmonic than candidates (a) and (b).

(27) After 7000 Tokens of Seed Data

		bejt, PRET, SG		*Superheavy	RM:pret	*[(CV)$_i$(CV)$_i$]	*[a]	Dep[+long]-IO	MaxV-IO	Dep[+back]-IO	Ident[back]-BR	Integrity-IO	$\mathscr{H}$
		Ranking Value	200	157.7	156.9	104.8	100.7	99.8	97.6	94.5	92.7	
a.	['bebajt] ⇒ /RED$_{PRET}$, bejt/					1			1	1	2	-3.4399×10^{45}
b.	☞ [bajt] ⇒ /bejt, ∅$_{PRET}$/					1			1			-3.4398×10^{45}
c.	['bebejt] ⇒ /RED$_{PRET}$, bejt/			1							2	-4.68×10^{68}
d.	['bejt] ⇒ /RED$_{PRET}$, bejt/		1					1				-2.258×10^{68}
e.	['bejt] ⇒ /bejt, ∅$_{PRET}$/		1									-2.258×10^{68}
f.	['babajt] ⇒ /RED$_{PRET}$, bejt/			1	2			2			2	-4.68×10^{68}
g.	['be:jt] ⇒ /bejt, ∅$_{PRET}$/	1				1						-7.226×10^{86}

- When *[a] has less impact, the violations of Ident[back]-BR and Integrity diminish the likelihood of candidate a.

(28) After 25000 Tokens of Seed Data

	bejt, PRET, SG		*SUPERHEAVY	RM:PRET	*[(CV)i(CV)i]	DEP[+long]-IO	*[a]	MAXV-IO	DEP[+back]-IO	IDENT[back]-BR	INTEGRITY-IO	$\mathscr{H}$
	Ranking Value	200	157.7	156.9	102.1	102	101.9	97.6	63.7	35.7	
a. ☞	['bebajt] ⇒ /RED$_{PRET}$, bejt/					1		1	1	2	-1.9×10^{44}
b. ☞	[bajt] ⇒ /bejt, ∅$_{PRET}$/					1		1			-1.9×10^{44}
c.	['bebejt] ⇒ /RED$_{PRET}$, bejt/			1						2	-4.6×10^{68}
d.	['bejt] ⇒ /RED$_{PRET}$, bejt/		1				1				-2.2×10^{68}
e.	['bejt] ⇒ /bejt, ∅$_{PRET}$/		1								-2.2×10^{68}
f.	['babajt] ⇒ /RED$_{PRET}$, bejt/			1		2		2		2	-4.68×10^{68}
g.	['be:jt] ⇒ /bejt, ∅$_{PRET}$/	1			1						-7.23×10^{86}

- The violations of IDENT[back]-BR and INTEGRITY insufficiently penalize candidate (a), rendering the differences in likelihood between candidates (a) and (b) negligible.

5.3. See **Appendix** for violation profiles and impact of seed training data on the other tableaux in the simulation.

5.4. Example Learning Path: Training on 12000 Adult Seed Data Tokens

Token #	INTEGRITY	MAX[+back]-IO	DEP[+back]-IO	DEP[-back]-IO	*[a]	IDENT[back]-BR	DEP[+long]-IO
Initial state	100	100	100	100	200	100	100
1000	98.12	100.93	98.28	101.09	169.86	98.15	99.65
2000	97.98	101.15	97.57	102.1	149.45	97.86	99.65
3000	97.7	101.15	97.57	102.836	139.78	98.07	99.65
4000	96.8	101.15	97.57	103.54	130.16	97.63	99.65
5000	95.91	101.15	97.57	104.38	119.17	96.7	99.65
6000	95.17	101.15	97.57	105.29	108.46	96.11	99.65
7000	92.8	101.15	97.57	105.34	105.29	94.75	99.81
8000	89.78	101.15	97.57	105.15	104.87	93.09	100.014
9000	86.9	101.15	97.57	104.9	104.78	91.65	100.24
10000	83.84	101.15	97.57	104.68	104.71	90.03	100.48
11000	80.65	101.15	97.57	104.46	104.37	88.09	100.69
12000	77.58	101.15	97.57	104.19	104.22	86.35	100.941

Token #	MAXC-IO	MAXV-IO	RM:PRET	*[(CV)i(CV)i]	OCP′	*SUPERHEAVY
Initial state	100	100	0	0	200	200
1000	99.61	99.3	119.38	118.77	200	200
2000	99.65	99.62	156.62	156.59	200	200
3000	99.65	99.79	156.62	156.59	200	200
4000	99.65	100.04	156.624	156.59	200	200
5000	99.65	100.3	156.62	156.59	200	200
6000	99.65	100.46	156.62	156.59	200	200
7000	99.49	100.57	156.62	156.59	200	200
8000	99.28	100.59	156.62	156.59	200	200
9000	99.05	100.62	156.62	156.59	200	200
10000	98.81	100.61	156.62	156.59	200	200
11000	98.6	100.65	156.62	156.59	200	200
12000	98.35	100.69	156.62	156.59	200	200

5.5. Generalizations on "Adult" Learning Trajectory

5.5.1. Weights of the faithfulness constraints INTEGRITY, DEP[+back]-IO, and IDENT[back]-BR fall.

- INTEGRITY and IDENT[back]-BR are violated by all best parses of the training data on preterite forms.
- DEP[+back]-IO is violated by the relatively frequent Class I–V preterites singular.

5.5.2. Weights of the faithfulness constraints MAX[+back]-IO and DEP[-back]-IO rise.

- These two constraints penalize vowel-changing alternatives in Class VI and VII forms. They conspire to ensure that underlying /a/ is preserved.

5.5.3. Weights of the morphological or "unnatural" constraints RM:PRET and *[$(CV)_i(CV)_i$] rise.

- It is essential for the weights of these constraints to rise sufficiently during the seed training period in order to move the grammar away from its initial winners, which do not give phonological expression to the morphosyntactic feature PRETERITE.

5.5.4. Weight of the markedness constraint *[a] falls.

- However, should the weight of this constraint fall too low, it risks permitting the mapping ['bajt] ⇒ /bejt, PRES/, with the result that incorrect present tense forms would be predicted to exist.

5.6. Learner's Outputs after 12000 Adult Seed Data Tokens on Five Agents

(29) Agent 1:

Input	**Current Winner**	Diachronically Correct?
|bejt, PRES|	['bejt] → /bejt, PRES/	Yes
|dab, PRES|	['dedab] → /dab, PRES/	**No**
|bejt, PRET, SG|	['bajt] → /bejt, $\varnothing_{PRET}$/	Yes
|bejt, PRET, PL|	['bebit] → /bejt, $\varnothing_{PRET}$, PL/	Yes
|ber, PRET, SG|	['bebar] → /ber, $\varnothing_{PRET}$/	Yes
|ber, PRET, PL|	['be:r] → /RED_{pret}, ber, PL/	**No**
|dab, PRET, SG|	['deb] → /dab, $\varnothing_{PRET}$/ Or /RED_{PRET}, dab/	**No**
|dab, PRET, PL|	['de:b] → /RED_{PRET}, dab, PL/	**No**
|hajt, PRET, SG|	['hehajt] → /RED_{PRET}, hajt/	Yes
|hajt, PRET, PL|	['hehit] → /RED_{PRET}, hajt, PL/	**No**

(30) Agent 2:

Input	**Current Winner**	Diachronically Correct?
|bejt, PRES|	['bejt] → /bejt, PRES/	Yes
|dab, PRES|	['dedab] → /dab, PRES/	Yes
|bejt, PRET, SG|	['bajt] → /bejt, $\varnothing_{PRET}$/	Yes
|bejt, PRET, PL|	['bebit] → /bejt, $\varnothing_{PRET}$, PL/	Yes
|ber, PRET, SG|	['bebar] → /ber, $\varnothing_{PRET}$/	Yes
|ber, PRET, PL|	['be:r] → /RED_{pret}, ber, PL/	**No**
|dab, PRET, SG|	['da:b] → /dab, $\varnothing_{PRET}$/	Yes
|dab, PRET, PL|	['da:b] → /dab, $\varnothing_{PRET}$, PL/	Yes
|hajt, PRET, SG|	['hehajt] → /RED_{PRET}, hajt/	Yes
|hajt, PRET, PL|	['hehit] → /RED_{PRET}, hajt, PL/	**No**

(31) Agents 3 and 5:

Input	**Current Winner**	Diachronically Correct?
|bejt, PRES|	['bejt] → /bejt, PRES/	Yes
|dab, PRES|	['dedab] → /dab, PRES/	Yes
|bejt, PRET, SG|	['bajt] → /bejt, $\varnothing_{\text{PRET}}$/	Yes
|bejt, PRET, PL|	['bebit] → /bejt, $\varnothing_{\text{PRET}}$, PL/	Yes
|ber, PRET, SG|	['bebar] → /ber, $\varnothing_{\text{PRET}}$/	Yes
|ber, PRET, PL|	['be:r] → /ber, $\varnothing_{\text{PRET}}$/	Yes
|dab, PRET, SG|	['da:b] → /dab, $\varnothing_{\text{PRET}}$/	Yes
|dab, PRET, PL|	['da:b] → /dab, $\varnothing_{\text{PRET}}$/	Yes
|hajt, PRET, SG|	['hehajt] → /RED_{PRET}, hajt/	Yes
|hajt, PRET, PL|	['hehit] → /RED_{PRET}, hajt, PL/	**No**

(32) Agent 4:

Input	**Current Winner**	Diachronically Correct?
|bejt, PRES|	['bejt] → /bejt, PRES/	Yes
|dab, PRES|	['dedab] → /dab, PRES/	Yes
|bejt, PRET, SG|	['bajt] → /bejt, $\varnothing_{\text{PRET}}$/	Yes
|bejt, PRET, PL|	['bebit] → /bejt, $\varnothing_{\text{PRET}}$, PL/	Yes
|ber, PRET, SG|	['bebar] → /ber, $\varnothing_{\text{PRET}}$/	Yes
|ber, PRET, PL|	['be:r] → /RED_{pret}, ber/	**No**
|dab, PRET, SG|	['da:b] → /dab, $\varnothing_{\text{PRET}}$/	Yes
|dab, PRET, PL|	['de:b] → /RED_{PRET}, dab/	No
|hajt, PRET, SG|	['hehajt] → /RED_{PRET}, hajt/	Yes
|hajt, PRET, PL|	['hehit] → /RED_{PRET}, hajt, PL/	**No**

5.6.1. These four patterns exhaust the set of winners arrived at after training on 12000 adult tokens.

5.6.2. Observations:

- The non-reduplicated option for the singular of Classes I–V and the plural of Classes I–III is preferred by all agents. This is the effect of harmonic bounding at work.
- The Class IV plural always arrives to the correct SR, but not always the correct UR. This results when the weight of DEP[+long]-IO remains too high.
- The Class VI singular forms are rather variable: [da:b], [de:b], and [deb] are all possible winners.

5.6.3. On 10000 agents, the pattern of Agent 3/5 occurs in 2722 agents – slightly more often that any of the other patterns. We infer from this result that the frequency distributions in the seed data tend to result in grammars nearly effectuating the desired diachronic changes.

5.7. Learner's Output after 1000000 Adult Seed Data Tokens

5.7.1. Convergence will never occur – constraint weights never stabilize and cease to update.

5.7.2. Thus: a stable grammar with a consistent set of outputs cannot be learned from training on the adult seed data alone.

6. Simulation Walk-Through: Agent Learning

6.1. The agents are now trained on the current winning surface representations of each other agent (including itself) after the initial training on seed data.

- The number of tokens from each each is equal to $\frac{\text{initial number of seed data tokens}}{\text{number of agents}}$.

- Following the agent-based training in Generation 1, the agents are also trained on a final smaller batch of seed data, where the number of tokens is also equal to $\frac{\text{initial number of seed data tokens}}{\text{number of agents}}$.

6.2. When the winning SRs contain unreduplicated forms like ['bajt] and ['bit], the weights of INTEGRITY and IDENT[back]-BR decrease less in weight. Note that the weights of these two constraints have decreased little with respect to the amount of decrease seen in the initial seed data training.

(33) Example Rankings for one agent after full training in Generation 1

Constraint	Ranking Value
*SUPERHEAVY	200
OCP′	200
RM:PRET	157.452
*[(CV)$_i$(CV)$_i$]	157.277
DEP[-back]-IO	102.271
DEP[+long]-IO	102.261
MAX[+back]-IO	100.420
MAX-IO	99.341
DEP[+back-IO]	89.923
IDENT[back]-BR	82.043
INTEGRITY-IO	69.206
*[a]	48.972

6.3. When most or all of the agents from one generation have selected winners with SRs that reflect the desired Gothic forms, the agents of the subsequent generation learn constraint weights that best fit those desired winners, and similar weights are seen across generations.

(34) Example Rankings for Three Agents in Generations 2–4 (in a simulation where Gothic forms appear in Generation 2)

Constraint	Ranking Value: Gen. 2	Ranking Value: Gen. 3	Ranking Value: Gen. 4
*SUPERHEAVY	200	200	200
OCP′	200	200	200
*[(CV)$_i$(CV)$_i$]	139.661	143.329	146.165
RM:PRET	139.047	142.926	145.814
DEP[-back]-IO	104.826	104.890	104.702
MAX[+back]-IO	104.116	104.140	104.236
INTEGRITY	103.480	103.519	103.581
MAXV	100.777	100.427	100.701
IDENT[back]-BR	99.807	99.903	100.015
MAXC	99.661	99.605	99.669
DEP[+long]	98.708	98.823	98.732
DEP[+back]	85.018	84.002	83.814
*[a]	-13.084	-21.403	-25.396

- NB: because the ranking values used in calculating winners are $e^{\text{ranking value}}$, negative ranking values indicate very small weights, not true negative weights.

7. Testing the Learning Parameters

7.1. In these learning simulations, the following *parameters* potentially impact the learning pathway:

7.1.1. the number of **agents** per generation. This parameter impacts the relative influence of any individual agent within and across generations; fewer agents = greater impact.
E.g., if only two agents are used, and both are of the Agent 1 type after training on the seed data, arriving to the Gothic grammar may require more generations.

7.1.2. the number of **training tokens** on initial **seed data**. This parameter determines how influential the Early Proto-Germanic forms will be in the course of simulation.

7.1.3. the **global plasticity multiplier** (cf. 5.5.6). This parameter determines the amount by which the learner will update constraint weights in the face of a training datum.

7.2. While the **number of learning generations** used in a simulation is set by the user, this number does not itself impact the global outcome, except insofar as the number of generations is sufficient to reach a stable terminal state, given the other parameters above.

7.3. Simulations testing global plasticity and number of initial seed data tokens.

- Cells in the table indicate the number of learning generations needed for all agents to arrive to the Gothic SRs and URs. All simulations used five agents.

(35) Plasticity

Seed Data Tokens/Global Plasticity	0.001	0.01	0.1	1	10
7000	Never	Never	2	4	6
12000	Never	Never	2	5–6	10
18000	Never	Never	4	6	12
25000	Never	Never	4–5	6	13
50000	Never	4–5	4–5	6	15
100000	Never	1	6	7	>20

7.4. When global plasticity is too low, the morphological constraints are not promoted quickly enough to take effect.

- Without the effect of RM:PRET the grammar prefers unmarked forms, that are then stably transmitted.
- Plasticity at 0.001 stabilizes on a grammar with unmarked preterite forms in Classes I–V.

7.5. Overall, there is a trade-off between global plasticity and number of seed data tokens:

- Higher plasticity brings the morphological constraints into play, but also demotes faithfulness constraints more quickly.
- Larger numbers of initial seed data tokens produce a Generation 1 that hews more closely to the seed data (cf. 6.8 above).
- In fact, when plasticity is too high, (1 or greater) the problem becomes that there is too little penalty for certain frequently violated constraints (especially *[a]), and present tense forms behave incorrectly.

7.6. A plasticity of 0.1 with initial training on 12000–50000 tokens is most plausible.

8. Summary and Conclusions

8.1. **Our Question:** are there conditions under which the *observed* winners are *not learned*, and instead replaced by new output forms under a (possibly) new grammar?

8.1.1. Yes, provided that the winners are either/or...

a. ...harmonically bounded by other candidates. This is the case for the singular forms of Classes I–V (['bebajt] ⇒ /RED$_{\text{PRET}}$, bejt/ vs. ['bajt] ⇒ /bejt, ∅$_{\text{PRET}}$/).

b. ...substantially worse than some other candidates when the grammar is in its initial state. This is the case for the plural forms of Classes IV and V (['be:r] ⇒ /RED$_{\text{PRET}}$, ber, PL/ vs. ['be:r ⇒ /ber, ∅$_{\text{PRET}}$, PL/).

8.1.2. Furthermore, learning must not be overly slavish: learning agents should not learning exclusively from "adult" grammars before learning from their own "imperfect" grammars. Agent-based learning is crucial to diachronic change.

8.1.3. Learning with respect to Faithfulness constraints must not be too rapid: if Faithfulness constraints are demoted too quickly, gratuitous violations will be licensed too readily.

8.2. In effect: learning is seeking a path to a categorical winners are not harmonically bounded.

8.2.1. From an initial state in which one candidate is always worse, when faced with categorical data, the best outcome is an even match.

8.2.2. From that same initial state, over the course of generations, the natural bias in favor of the more harmonic candidate will tend to give a greater proportion of the winners to the more harmonic candidate.

8.3. The open question is: does this sort of restructuring tend to occur relatively quickly, or over long periods of time, with substantial variation?

8.3.1. Diachronic syntax: variation arising from ambiguous parses seems to produce lengthy periods of structured variation (Kroch 1989).

8.3.2. Prosodic phonology: irrecoverable URs seem to result in nearly instant (single generation) restructuring (Bowers (2015) on Odawa).

8.3.3. A rational UR to account for the data is not irrecoverable and the problem does not lie in ambiguous parsing. Rather, the problem is that these winners have gratuitous phonological Faithfulness violations.

References

Albright, Adam, and Bruce Hayes. 2003. Rules vs. Analogy in English Past Tenses: A Computational/Experimental Study. *Cognition* 90.119–61.

Boersma, Paul, and Bruce Hayes. 2001. Empirical Tests of the Gradual Learning Algorithm. *Linguistic Inquiry* 42.45–86.

Boersma, Paul, and Joe Pater. 2013. Convergence Properties of a Gradual Learning Algorithm for Harmonic Grammar. `http://www.fon.hum.uva.nl/paul/papers/SGAproof71.pdf`.

Boersma, Paul, and David Weenink. 1992–2014. Praat. Doing Phonetics by Computer. Software. URL `www.praat.org`.

Bowers, Dustin. 2015. A System for Morphophonological Learning and its Consequences for Language Change. Ph.D. diss., University of California, Los Angeles.

Casali, Roderic F. 1996. Resolving Hiatus. Ph.D. diss., University of California, Los Angeles.

Goldwater, Sharon, and Mark Johnson. 2003. Learning OT Constraint Rankings Using a Maximum Entropy Model. In Jennifer Spenader, Anders Eriksson and Östen Dahl (eds.), *Proceedings of the Stockholm Workshop on Variation within Optimality Theory*, 111–20. Stockholm University Department of Linguistics.

Halle, Morris. 1997. On Stress and Accent in Indo-European. *Language* 73.275–313.

Hayes, Bruce. 2004. Phonological Acquisition in Optimality Theory: The Early Stages. In René Kager, Joe Pater and Wim Zonneveld (eds.), *Constraints in Phonological Acquisition*, 158–203. Cambridge: Cambridge University Press.

Hayes, Bruce, and Colin Wilson. 2008. A Maximum Entropy Model of Phonotactics and Phonotactic Learning. *Linguistic Inquiry* 39.379–440.

Hayes, Bruce, Kie Zuraw, Péter Siptár, and Zsuzsa Cziráky Londe. 2009. Natural and Unnatural Constraints in Hungarian Vowel Harmony. *Language* 85.822–63.

Jäger, Gerhard. 2007. Maximum Entropy Models and Stochastic Optimality Theory. In Jane Grimshaw, J. Maling, C. Manning, J. Simpson and A. Zaenen (eds.), *Architectures, Rules, and Preferences: A Festschrift for Joan Bresnan*, 467–79. Stanford: CSLI Publications.

Jarosz, Gaja. 2013. Learning with Hidden Structure in Optimality Theory and Harmonic Grammar: Beyond Robust Interpretive Parsing. *Phonology* 30.27–71.

Jesney, Karen, and Anne-Michelle Tessier. 2011. Biases in Harmonic Grammar: The Road to Restrictive Learning. *Natural Language and Linguistic Theory* 29.251–90.

Johnson, Keith. 2011. Modeling Phonology in Time. *UC Berkeley Phonology Lab Annual Report* 183–88.

Kiparsky, Paul. 2010. Compositional vs. Paradigmatic Approaches to Accent and Ablaut. In Stephanie W. Jamison, H. Craig Melchert and Brent Vine (eds.), *Proceedings of the 21st UCLA Indo-European Conference*, 137–82. Bremen: Hempen.

———. Forthcoming. Accent and Ablaut. In Michael Weiss and Andrew Garrett (eds.), *Handbook of Indo-European Studies*. Oxford: Oxford University Press.

Kroch, Anthony. 1989. Reflexes of Grammar in Patterns of Language Change. *Language Variation and Change* 1.199–244.

Kurisu, Kazutaka. 2001. The Phonology of Morpheme Realization. Ph.D. diss., University of California, Santa Cruz.

Pater, Joe. 2009. Learning in Praat: The Basics. `http://people.umass.edu/pater/praat-learning-guide-jp.pdf`.

Pater, Joe, Robert Staubs, and Coral Hughto. 2014. Grammatical Agent-Based Modeling of Typology. URL `http://blogs.umass.edu/pater/files/2011/10/pater-staubs-hughto-bootphon.pdf`, presentation. Modeling Group Meeting, December 11, 2014. LSCP, Paris.

Pinker, Steven, and Alan Prince. 1992. Regular and Irregular Morphology and the Psychological Status of Rules of Grammar. In *Proceedings of the 17th Annual Meeting of the Berkeley Linguistics Society*, 230–51. Berkeley, CA: Berkeley Linguistics Society.

Sandell, Ryan. 2015. Obligatory Contour Principle Effects in Indo-European Phonology: Statistical Evidence and the Morphology-Phonology Interface. In Stephanie W. Jamison, H. Craig Melchert and Brent Vine (eds.), *Proceedings of the 26th Annual UCLA Indo-European Conference*. Bremen: Hempen.

Schumacher, Stefan. 2005. 'Langvokalische Perfekta' in indogermanischen Einzelsprachen und ihr grundsprachlicher Hintergrund. In Gerhard Meiser and Olav Hackstein (eds.), *Sprachkontakt und Sprachwandel. Akten der XI. Fachtagung der Indogermanischen Gesellschaft, 17-23. September 2000, Halle an der Salle*, 591–626. Wiesbaden: Reichert.

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

Zukoff, Sam. 2014. On the Origins of Attic Reduplication. In Stephanie W. Jamison, H. Craig Melchert and Brent Vine (eds.), *Proceedings of the 25th UCLA Indo-European Conference*, 257–78. Bremen: Hempen.

————. 2015. Poorly-Cued Repetition Avoidance in Indo-European Reduplication. URL `https://mit.academia.edu/SamZukoff`, Presentation. 89th Annual Meeting of the Linguistic Society of America, January 9, 2015.

————. forthcoming. The Reduplication System of Ancient Greek and a New Analysis of Attic Reduplication. *Linguistic Inquiry* 48(3).

————. in progress. Indo-European Reduplication: Synchrony, Diachrony, and Theory. PhD Dissertation, MIT, Cambridge, MA.

Zukoff, Sam, and Ryan Sandell. 2015. The Phonology of Morpheme Realization in the Germanic Strong Preterites. In Thuy Bui and Deniz Özyıldız (eds.), *Proceedings of the 45th Annual North East Linguistics Society Conference*. Amherst, MA: GLSA.

1. Appendix: Violation Profiles of Preterite Forms

1.1. Class I–III Plurals: Seed Training Datum: ['bebit]

(36) Tableau for Class I–III Plurals

		|bejt, PRET, PL|	RM:PRET	*[(CV)$_i$(CV)$_i$]	INTEGRITY	DEP[+back]	*[a]	MAXV
a.	☞	['bebit] ⇒ /RED$_{PRET}$, bejt/			2			1
b.	>>	[bit] ⇒ /bejt, ∅$_{PRET}$/						1
c.		['bebejt] ⇒ /RED$_{PRET}$, bejt/		1	2			
d.		['bejt] ⇒ /RED$_{PRET}$, bejt/	1					1
e.		['bejt] ⇒ /bejt, ∅$_{PRET}$/	1					
f.		['babit] ⇒ /RED$_{PRET}$, bejt/			2	1	1	

1.1.1. Initial winner: ['bejt] ⇒ /bejt, ∅$_{PRET}$/

1.1.2. Training datum ['bebajt] maps best to ['bebit] ⇒ /RED, bejt/.

- Results in demotion of: INTEGRITY, MAXV
- Results in promotion of: RM:PRET

1.1.3. When the weights of INTEGRITY and MAXV are sufficiently low, also results in promotion of *[(CV)$_i$(CV)$_i$].

1.2. **Class IV–V Plurals:** Seed Training Datum: ['be:r]

(37) Tableau for Class IV–V Plurals

	|ber, PRET, PL|	RM:PRET	OCP'	INTEGRITY	DEP[+long]	MAXC	MAXV
a. ☞	['be:r] ⇒ /RED$_{PRET}$, ber/			2		1	1
b. >>	['be:r] ⇒ /ber, ∅$_{PRET}$/				1		
c.	['bebr] ⇒ /RED$_{PRET}$, ber/		1	2			1
d.	['ber] ⇒ /RED$_{PRET}$, ber/	1				1	1
e.	['ber] ⇒ /ber, ∅$_{PRET}$/	1					

1.2.1. Training datum ['be:r] maps best to ['be:r] ⇒ /ber, ∅$_{PRET}$/.

- Results in demotion of: DEP[+long]
- Results in promotion of: RM:PRET

1.2.2. The ranking RM:PRET ≫ DEP[+long] is sufficient to result in a stable SR ⇒ UR mapping.

1.3. **Class VI Singular:** Seed Training Datum: ['dedab]

(38) Tableau for Class VI Singular

	|dab, PRET, SG|	RM:PRET	*[(CV)$_i$(CV)$_i$]	INTEGRITY	DEP[-back]	MAX[+back]	IDENT[back]-BR	*[a]	MAXV	DEP[+long]
a. ☞	['dedab] ⇒ /RED$_{PRET}$, dab/			2	1		1	1		
b. >>	[da:b] ⇒ /dab, ∅$_{PRET}$/							1		1
c.	['dadab] ⇒ /RED$_{PRET}$, dab/		1	2				2		
d.	['dab] ⇒ /RED$_{PRET}$, dab/	1						1	1	
e.	['dab] ⇒ /dab, ∅$_{PRET}$/	1						1		
f.	['deb] ⇒ /dab, ∅$_{PRET}$/				1	1				

1.3.1. Initial winner: ['deb] ⇒ /dab, ∅$_{PRET}$/ (*[a] ≫ DEP[-back])

1.3.2. Training datum ['dedab] maps best to ['dedab] ⇒ /RED, dab/.

- Results in demotion of: INTEGRITY, IDENT[back]-BR
- Results in promotion of: MAX[+back]

1.3.3. When the weights of INTEGRITY and *[a] are sufficiently low, also results in promotion of *[(CV)$_i$(CV)$_i$].

1.4. **Class VI Plural:** Seed Training Datum: ['de:b]

(39) Tableau for Class VI Plural

	|dab, PRET, PL|	RM:PRET	*[(CV)$_i$(CV)$_i$]	INTEGRITY	DEP[-back]	MAX[+back]	IDENT[back]-BR	*[a]	MAXV	DEP[+long]	OCP
a.	☞ ['de:b] ⇒ /RED$_{PRET}$, dab/			2	1		1	1			
b.	>> [da:b] ⇒ /dab, ∅$_{PRET}$/							1		1	
c.	['dedab] ⇒ /RED$_{PRET}$, dab/			2				1			
d.	['dadab] ⇒ /RED$_{PRET}$, dab/		1	2				2			
e.	['deb] ⇒ /dab, ∅$_{PRET}$/	1						1			
f.	['de:b] ⇒ /dab, ∅$_{PRET}$/				1	1				1	
g.	['dedb] ⇒ /RED$_{PRET}$, dab/			2	1			1			1

1.4.1. Initial winner: ['deb] ⇒ /dab, ∅$_{PRET}$/ (due to initial ranking *[a] ≫ DEP[-back])

1.4.2. Training datum ['deb] maps best to ['dedab] ⇒ /RED, dab/.

- Results in demotion of: DEP[+long]
- Results in promotion of: *[(CV)$_i$(CV)$_i$]

1.4.3. When the weights of INTEGRITY and*[a] are sufficiently low, also results in promotion of *[(CV)$_i$(CV)$_i$].

1.5. **Class VII Singular:** Seed Training Datum: ['hehajt]

(40) Tableau for Class VII Singular

	|hajt, PRET, SG|	RM:PRET	*[(CV)$_i$(CV)$_i$]	INTEGRITY	DEP[-back]	MAX[+back]	IDENT[back]-BR	*[a]	MAXV	DEP[+long]	*SUPERHEAVY
a.	☞, >> ['hehajt] ⇒ /RED$_{PRET}$, hajt/			2	1		1	1			
b.	[hahajt] ⇒ /RED$_{PRET}$, hajt /		1	2				2			
c.	['hehejt] ⇒ /RED$_{PRET}$, hajt/		1	2	1	1					
d.	['hajt] ⇒ /RED$_{PRET}$, hajt/	1						1	1		
e.	['hajt] ⇒ /hajt, ∅$_{PRET}$/	1						1			
f.	['ha:jt] ⇒ /hajt, ∅$_{PRET}$/									1	1

1.5.1. Initial winner: ['hajt] ⇒ /hajt, ∅$_{PRET}$/

1.5.2. Training datum ['hehajt] maps best to ['dedab] ⇒ /RED, dab/.

- Results in demotion of: *[a], IDENT[back]-BR, INTEGRITY
- Results in promotion of: *[(CV)$_i$(CV)$_i$], DEP[-back], MAX[+back]

1.6. **Class VII Plural:** Seed Training Datum: ['hehit]

(41) Tableau for Class VII Plural

	|hajt, PRET, PL|	RM:PRET	*[(CV)$_i$(CV)$_i$]	INTEGRITY	DEP[-back]	MAX[+back]	IDENT[back]-BR	*[a]	MAXV	DEP[+long]	*SUPERHEAVY
a.	☞ ['hehit] ⇒ /RED$_{PRET}$, hajt/			2			1	1	1		
b.	>> ['hehajt] ⇒ /RED$_{PRET}$, hajt/			2	1		1				
c.	[hahajt] ⇒ /RED$_{PRET}$, hajt /		1	2				2			
d.	['hajt] ⇒ /RED$_{PRET}$, hajt/	1						1	1		
e.	['hajt] ⇒ /hajt, ∅$_{PRET}$/	1						1			
f.	['ha:jt] ⇒ /hajt, ∅$_{PRET}$/									1	1
g.	['hehejt] ⇒ /RED$_{PRET}$, hajt/		1	2	1	1					

1.6.1. Initial winner: [ˈhajt] ⇒ /hajt, $\varnothing_{\text{PRET}}$/

1.6.2. Training datum [ˈhehajt] maps best to [ˈhehajt] ⇒ /RED, hajt/.

- Results in demotion of: *[a], IDENT[back]-BR, INTEGRITY
- Results in promotion of: *[$(CV)_i(CV)_i$], DEP[-back], MAX[+back]