

# Recursion In Humpback Whale Song

Linguistics Tripos  
Candidate number 9563B

Word count: 9776

## Table of contents

1. Introduction and roadmap	4
2. Background on humpback song	5
2.1 Function and context	5
2.2 Phonetics	5
2.3 Why do we need hierarchy?	7
3. Data	9
4. The architecture of humpback syntax	10
4.1 Is one phrasal layer enough?	10
4.2 Confusion around the nature of themes: repeating strings or alternations?	12
4.3 Sisterhood as alternation	14
4.4 A derivation for themes	14
5. Computational implementations	17
5.1 Context-free grammars (CFGs)	17
5.1.1 A CFG implementation of Phrase C1	18
5.1.2 Generalizability of the grammar	21
5.1.2.1 Undergeneration	21
5.1.2.2 Overgeneration	23
5.1.3 Pairwise grammar induction	23
5.2 Finite-state automata	24
6. Discussion	28
6.1 Self-similarity and Recursion	28
6.2 Limits on depth of embedding	29
6.3 Physiological explanations?	31
6.4 Hybridization: an independent constituency test?	31
6.5 Syntax, Phonology and Beyond	32
6.6 The evolutionary picture	35
7. Future work	36

Acknowledgements	37
Appendix I: Derivation with $\text{Alt}(X)$	38
Appendix II: CFGs to PCFGs	38
Probabilistic context-free grammars (PCFGs)	38
Training	39
Appendix III: Induction	40
Appendix IV: Phrase A2	41
Software	43
Bibliography	43

# 1. Introduction and roadmap

The most salient structural quality of humpback song is the repetition of subroutines within the larger routine that is the entire song — as well as the repetition of the song itself. Payne and McVay (1971) use these patterns of *nested repetition* to posit a syntactic object smaller than the entire song but larger than its atomic sound units, called a *phrase*, which is the object of intra-song repetition. In so doing, they implicitly establish repeatability as a diagnostic of phrasehood: phrases are delineable from surrounding material because units internal to a phrase repeat together, and to the exclusion of other units: if a song consists of two adjacent phrases [AB][CD], the sequences ABAB and CDCD will be attested, and crucially, BCBC will *never* be attested, since it repeats elements that do not form a constituent. This constituency test parallels those used in linguistic syntax, like co-ordination and pro-form replacement, and gives us a means of falsifying posited constituents and proposing new ones.

After introducing the reader to humpback song (Section 2) and to the corpus used in this dissertation (Section 3), I will extend the reasoning above to *phrase-internal* structure, and show that the repetition patterns in one particularly complex phrase cannot be explained without multiply-embedded, binary-branching hierarchical structures, with a depth of three phrasal layers (Section 4). I propose a recursive derivation for humpback phrase structure that I believe to be the simplest model capable of generating all and only the attested variation. In Section 5, I support the generalizability of this analysis by implementing my proposed grammar as a context-free grammar (CFG), and show that this CFG is able to generate all the attested instances of the phrase being parsed, without overgenerating. I also discuss the empirical problems faced in modelling humpback syntax with simpler grammars, finite-state automata, which are unable to avoid either under- or overgenerating the attested variation, as a consequence of their inability to keep track of hierarchical information. In Section 6, I discuss the import of the extended hierarchical structure proposed here for theories of the evolution of syntax. I propose that humpbacks, like humans and unlike all other species that we know of, are capable of building syntax via the recursive operation Merge. I highlight that this capacity appears in a different cognitive context from human language syntax, and is subject to an entirely different linearization process, the details of which likely impose limitations on the depth of embedding we find in humpback phrase structure.

## 2. Background on humpback song

### 2.1 Function and context

The function of humpback song is the subject of an ongoing debate that shows no sign of letting up — or even converging on a handful of possibilities, for that matter<sup>1</sup>. Recent proposals have variously claimed their use for (and here I quote Frazer & Mercado (2000, p. 164) who themselves think the songs are sonar):

*“...courtship and pair bonding, sexual advertisement, maintenance of spacing between males, establishment of dominance, migratory beacons, synchronization of ovulation [of hearers<sup>2</sup>] and maintenance of contact”*

This dissertation is a formal study rather than a functional one, so I will address what I think to be a valid and self-contained question, disregarding functional considerations, namely: what is the minimally complex computational system needed to generate all and only the strings attested in the song of a given singer? Vital to this discussion is the fact that the songs are acquired, not innate. They vary widely both in phonetic detail and in structure, across time and space, much like human languages. Also like human languages, the songs are shared within populations, and subject to both gradual change and patterns of introduction and replacement: see Garland et al. (2011) for a record of the propagation of several successive songs across breeding groups in South Pacific during a decade-long period.

### 2.2 Phonetics

Humpback songs are produced by pushing air inhaled during surfacings between the singer’s lungs and the laryngeal sac (Adam et al, 2013). The positioning of the vocal folds along the trachea between these two air sources allows for bidirectional phonation, without air ever escaping the singer’s vocal tract. Bidirectional phonation has been invoked to explain some aspects of song structure (Mercado and Handel, 2015) — a proposal to which I will return in Section 6.3.

---

<sup>1</sup> Not that it should, necessarily. Berwick and Chomsky (2015) summarize Lewontin’s (2001) remarks with regards to bones as follows: *“bones do not have a single, unambiguous “function.” While it is true that bones support the body, allowing us to stand up and walk, they are also a storehouse for calcium and bone marrow for producing new red blood cells, so they are in a sense part of the circulatory system. What is true for bones is also true for human language.”* The same could be true of humpback song.

<sup>2</sup> Only males sing — though females have similar phonetic abilities.

Acoustically, the songs are composed of discrete sounds (henceforth, *units*) separated from one another by silence. Thanks to these pauses, the segmentation problem that an alien would encounter faced with the human speech stream simply does not exist. As such, the primary locus of transcriptional discrepancy is not in the delineation of units, but in their classification.

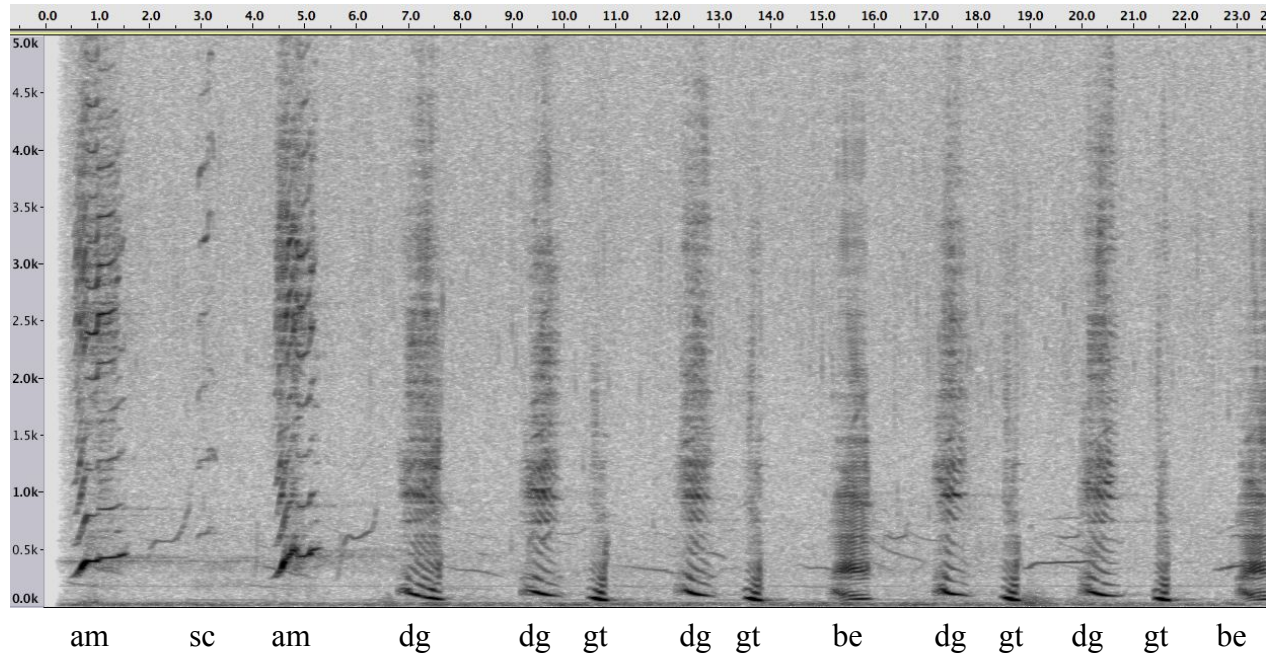


Figure 1. A spectrogram showing the first 23 seconds of the Song C recording used in this study (see Section 3).

Units are traditionally classified according to their duration, fundamental frequency ( $F_0$ ) contour, and average  $F_0$ , yielding a tripartite nomenclature as follows (the bolded letters are used to form acronyms):

Duration (optional)	$F_0$ contour	Average $F_0$ (from low to high)
short long	ascending descending n-shaped (rise-fall) u-shaped (fall-rise) modulated (periodic peaks)	<b>b</b> ellow (pulsative) <b>g</b> runt <b>g</b> roan <b>m</b> oan <b>c</b> ry <b>s</b> queak <b>sh</b> riek <b>t</b> rumpet (steep pitch rise)

Table 1. A rough schematic of the unit classification system used for transcription.

One source of noise in the data is background singing from nearby humpbacks. This is problematic when the background singer and primary singer are at similar distances from the hydrophone, but when the background singer is further away, the higher harmonics are disproportionately damped by the intervening water, making the units easy to weed out by ear or on a spectrogram. Consider Figure 2 in this connection, noting the absence of harmonics for the unit around 0:02, in contrast to the unit at 0:03, which is equally faint but harmonically richer, due to its proximity. Since the source of the former unit is clearly more distant than the primary singer, I exclude it from the transcription.

### 2.3 Why do we need hierarchy?

Pretheoretically, we can note that the strings of sound units in humpback song show patterns of *nested repetition*: that is, repetitions of long sequences that are themselves composed of shorter repeating sequences. The prevalence of this kind of pattern has received empirical support from an information-theoretic study by Suzuki, Buck, and Tyack (2006), who found statistically significant periodicities at two discrete levels: one at 6-8 units and one at 180-400 units. The lower periodicity corresponds to what Payne & McVay (1971) identify as the *phrase*, a unit in humpback song smaller than the song as a whole, but larger than the unit. Phrases are (definitionally) the object of intra-song repetition. The higher periodicity corresponds to whole-song repetition.

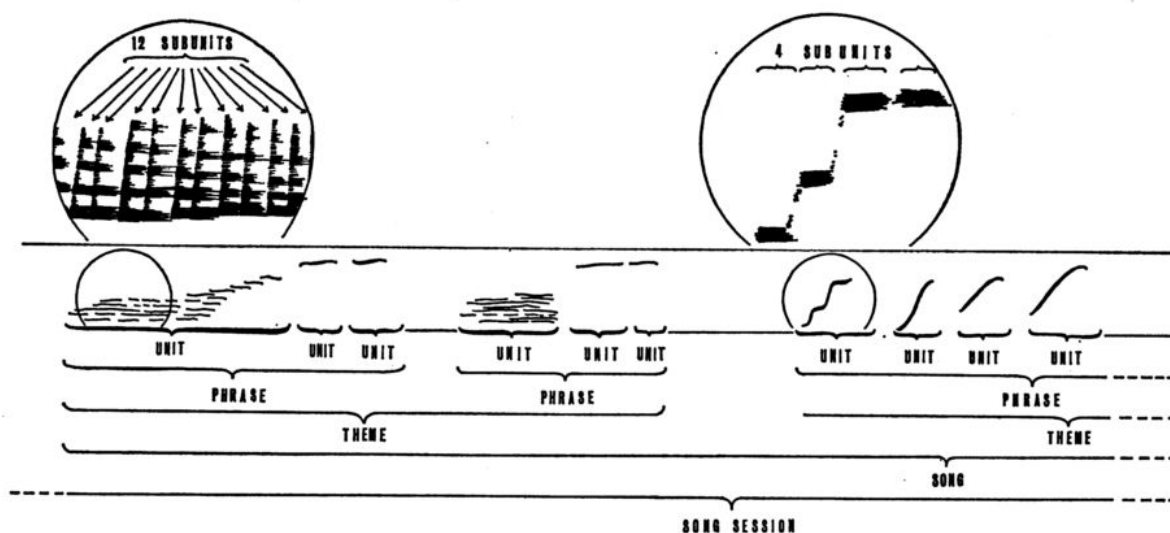


Figure 2. Payne and McVay's proposal for song structure.

Cholewiak, Sousa-Lima & Cerchio (2013) provide a good summary of the currently accepted terminology on page 316 of their discussion of current classification issues:

*“A set of units is combined to form a **phrase**. Similar phrases are repeated to form a **theme**. The **song** is defined as the combination of multiple distinct themes. A **song session** consists of a series of repeated **songs** with silent intervals of less than a minute...”* [italics replaced with bold]

The structure laid out here is explicitly one that interleaves combination with repetition: the relationship between units and phrases, and between themes and songs, is of the form  $X...Y > Z$  (combination), while the relationship between phrases and themes, and between songs and song sessions is of the form  $X...X > Z$  (repetition). Below I provide an idealized schematic of this structure, assuming two repetitions and binary combinations in all cases. Indices indicate identical syntactic objects, and as the reader can see, the four layers of branching structure here alternate between combining (branching to objects with different indices), and repeating (branching to objects with identical indices).

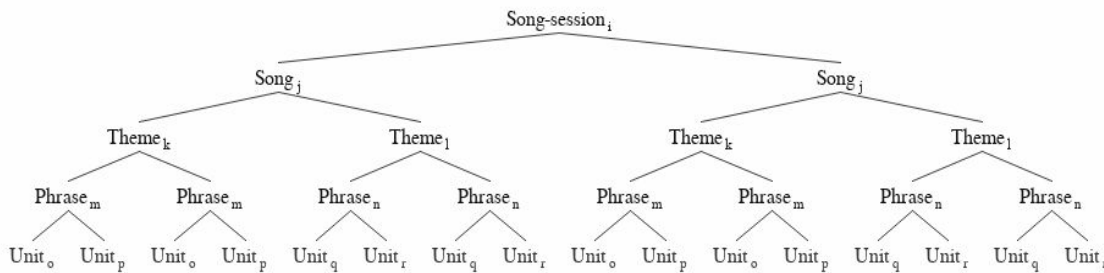


Figure 3. A tree schematization of the mainstream structure assigned to humpback song, which interleaves repetition with combination.

This interleaving is an indication that two things are really being represented here: hierarchical, combinatorial structure, on the one hand, and the repetition patterns through which this is diagnosed, on the other. While *theme* and *song session* are useful objects, it should be stressed that they are not levels of the combinatorial structure, in which songs dominate phrases, which in turn dominate units. To look at the combinatorial structure in isolation, we can abstract away the layers that do not combine two distinct objects. This gives us the following core hierarchical structure for humpback song:

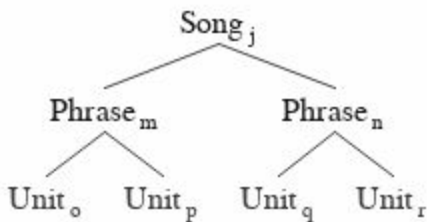


Figure 4. A schema of the **combinatorial** structure of humpback song.



The tree in Fig. 4 can be expanded into a tree that interleaves combination and repetition (Fig. 3) by combining each node with a copy of itself, from the top down. In other words, the content of a specific phrase resides in the layers in Fig. 4, while the additional expansion process (repeating phrases to form themes, etc.) is common to all phrases — after all, the nodes in Fig. 4 are *definitionally* repeatable. The exact details of the repetition — how many times different nodes get repeated — is another locus of idiosyncratic information, but is distinguishable from the hierarchical structure of individual phrases, which will be the primary subject of inquiry in this dissertation.

### 3. Data

In my analysis I will draw from a corpus of one Caribbean song type, which I will refer to as Song C. The data consists of one recording of a single singer made by Paul Knapp, Jr. off the coast of Tortola in the British Virgin Islands on February 14th, 1992. The recording is 25 minutes long, and contains 730 units, without pauses or interruptions. This song has been published as a track on Knapp's CD *Listening to Humpback Whales*, and is used in this project with his kind permission. I transcribed this recording myself for use in this dissertation, using the classification system in Section 2.2. In my transcription of Song C, I have assigned the units of this song the letters *a* through *e*, in order of their first appearance in the recording, as an aid to the reader. For the record, their traditional names would be roughly as follows:

1. a: Ascending moan (AM)
- b: Short cry (SC)
- c: Descending groan (DG)
- d: Grunt (GT)
- e: Bellow (BE)
- f: Squeak (SQ)
- g: Violin (V)
- h: Trumpet (TR)
- i: Descending Moan (DM)

My primary interest is in *phrase-internal* structure, and so I will refer throughout my analysis to one phrase in particular, which I call C1. I restrict myself in this way both for the sake of continuity and because C1 turns out to be of particular interest due to its internal repetition patterns, to be discussed further in section 4.5.<sup>3</sup> That being said, I will return to inter-phrasal structure in Section 6.1, and so I reproduce the entire Song C corpus below. It contains three repetitions of Song C, each containing the same five themes (repetitions of Phrases C1-C5).

---

<sup>3</sup> I apply a similar analysis to a phrase from a Polynesian song, *Song A*, in Appendix V.

Phrase	Units	First repetition	Second repetition	Third repetition
C1	a b c d e	a b a c c d c d e c d c d e c d c d e a a b a a c c d e c d c d e	c d c d e c d a a b a a c c d e c d c d e c d c d e a a b a a e c d c d e c d c d e c d c d e	c d c d e c d c d e c d c d e c d a a b a b a d c d c d e c d c d e c d a a a b a d c d e c d c d e c d c d e
C2	f g	fffffffffffffffffffff fffffffffffffg g fff ffffffffffffffffff fffffffffg g ffffff fffffffffffffg g ffff fffffg g fffffg g ff ffg g fffffg g fffffg g fffffg g	fffffffffffffffffffff fffffg g fffffffffff ffffffffffffffffffg g fff ffffffffffffffffffg g ffff fffffffffg g fffffffffff fg g fffffffffffg g fffffg g fffffg g	fffffffffffffffff fffffffffffffffff fg g fffffffffff fffffffffffffffff g g fffffffffff g g fffffffffff fffffg g ffffff fffffffffffffg fffffg g ffffff g g fffffg g ffff fg g fffffg g ff fffg g fffffg g
C3	h i	h h i i i i h h h i i i i i	h h i i i i i h h h i i i i i i	h h i i i i h h h i i i i h h h i i i i h h h i i i i i h h h i i i i i
C4	a i	a a a i i i i i a a a i i i i i a a a i i i i	a a a i i i i i a a a	a a a i i i i i a a a i i i i i i
C5	e i a b	e i i e i i e a a b a i i i e i i e i i e a a b a b a i i e i i e	i i i e i i e i i a a a i i i e i i e i i a a a i i i e i i e i i a a a i i i e i i i	—

Table 2. Caribbean corpus (song C). Read top-to-bottom, then left-to-right.

Taking after the schema in Fig. 4, we can preliminarily parse the structure of Song C as follows:

} Song level

} Phrase level

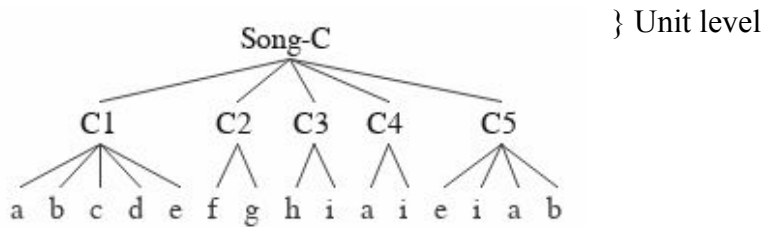


Figure 5. A preliminary parse of Song C, showing which units are contained in which phrases.

## 4. The architecture of humpback syntax

### 4.1 Is one phrasal layer enough?

The hierarchical structure posited by Payne and McVay (1971, henceforth P&M), and exemplified in Figures 3 and 4, is on the right track, but the single intermediate level they posit between the song and unit levels turns out to be insufficient to capture some of the more complex phrasal repetition patterns, which necessitate, at least in the case of Phrase C1, two additional phrasal levels. I will show this with respect to a single repetition of Phrase C1 (given in (2)). Recall from Table 2 that entire sequence repeats several times before moving on to the next theme.

(2) **aba** *cc d c d e c d c d e c d c d e*

The structure necessarily assumed for a phrase containing five units, in P&M and subsequent work, has flat quinary-branching structure beneath the phrase node, as shown below. Remember that these trees abstract away from repetition — so the terminal nodes won't replicate the string being modeled in full.

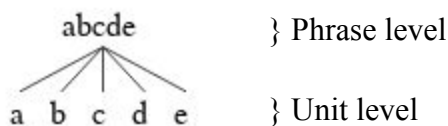


Figure 6. A quinary-branching parse of C1.

This flat structure means that we have no way of accounting for within-phrase repetition patterns. But note that the sequence from *c* to *e* repeats to the exclusion of the initial *aba* sequence, so if repetition is a diagnostic for phrasehood, we must posit a more complex branching structure, with a new phrasal level on which *ab* and *cde* are separate constituents.

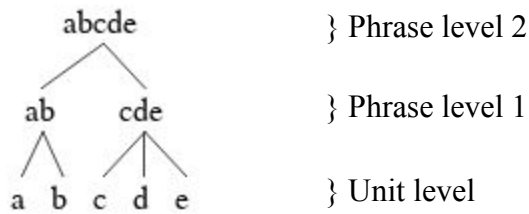


Figure 7. A binary-ternary parse of C1.

However, even this more complex binary-ternary structure fails to account for the repetition patterns *within* the new *cde* “subphrase”, since *cd* can repeat to the exclusion of *e*. If we add another sub-phrasal level embedded inside the *cde* subphrase, we can now fully explain all the repetition patterns within the larger phrase:

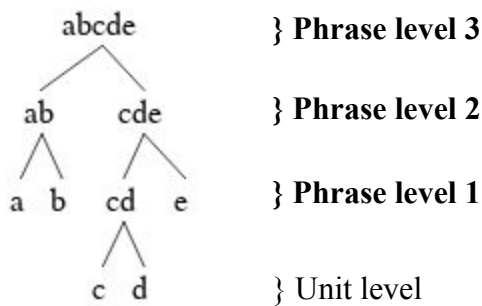


Figure 8. An entirely binary-branching parse of C1.

Now that our tree has a bijection between nodes and repeatable units, it creates loci for variability in the precise extent of repetition, which will be needed to account for other instantiations of phrase C1, as shown in the next section. I propose that the tree above, in conjunction with a generic process for expanding trees into strings is the simplest possible model of phrase C1 that does not overgenerate — since it allows for repetition within phrases, but rules out unattested repetitions of non-constituent unit combinations (like *ac*). However, as we shall see in the next section, we will need to think more carefully about what exactly the tree is encoding, if it is to be able to avoid undergenerating, and to be able to capture all the variability attested in our corpus.

## 4.2 Confusion around the nature of themes: repeating strings or alternations?

The analysis laid out in 4.1 is complicated when we move on to look at other instances of Phrase C1. For example, note that while in (2), the *cde* subphrase began with its *cd* component, which in turn began with *c*, this is not always the case. Consider the following strings, which show divergent transitions from the *ab* subphrase to the *cde* one:

- (3) ... a b **a** c d ...  
 (4) ... b a **a** e c d ...  
 (5) ... a b **a** d c d ...

To accommodate this variability in which node precedes which, one option is to rotate our tree like a mobile: in the first case, the *cde* node is rotated, while in the second case, the *cd* node is.

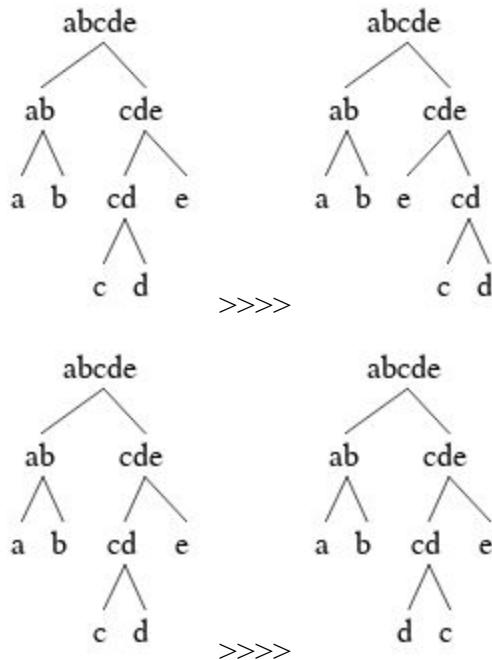


Figure 9. Two transformations of the tree in fig. 8.

But these transformations create as many problems as they solve, since the first predicts, for example, that *e* will never come at the end of the *cde* subphrase — but it does in (2). Cholewiak, Sousa-Lima & Cerchio (2013) attempt to provide guidelines for deciding which of the two components of a phrase comes first (emphasis mine):

*“... Phrases should be delineated in a way that **minimizes the occurrence of an incomplete phrase at the end** of a sequence of similar phrases (also called “hanging” phrases, consisting of only a portion of the repetitive structure, such as one subphrase).*

*... “**Transitional**” phrases combine units from two different phrase types (Payne and Payne 1985), usually an entire subphrase from the previous and subsequent themes. For example, using letters to indicate subphrases, in the phrase sequence:*

*ab ab ab **ad** cd cd cd”*

Note the use of *minimize* (line 1). By this the authors mean that no matter what choice is made — even if we side with the most common order — transcribers are often left to resort to either *hanging phrases* or *transitional phrases*, which are created ad-hoc to absorb unattached material at theme edges, in cases where adjacent themes begin and/or end with the wrong (unexpected) component. In this vein, the strings in (3) through (5) could be analyzed as containing either transitional or hanging phrases in between the *ab* subtheme and the *cd* subtheme.

But crucially, transitional “phrases” and hanging “phrases” fail our diagnostic for phrasehood, since they are *never repeated*. And since repetition patterns are the sole reason for positing hierarchical structure in the first place, calling these sequences “phrases” seem a clear violation of the terms of engagement. But without transitional phrases, how can we decide the ordering of elements within a phrase?

I propose that there is no right answer to this question, because it is the wrong question to ask (cf. the famous unanswerable question, “Have you stopped beating your wife?”). The question of phrase-internal precedence only arises if phrases encode precedence relations in the first place. If, on the other hand, phrases are reconstrued as *alternations*, the question is no longer on the table. I propose that themes expand phrases not by repeating them but by *alternating between* the unordered set of the phrase’s two daughters, and so variability in which member occurs at the edge of that alternation can be absorbed into the theme. I do not propose that this edge variability is random, but simply that this is not a property of the syntax, and instead of the variable linearization process that yields repetition patterns.

### 4.3 Sisterhood as alternation

Given the variability at theme edges discussed in 4.2, I propose that we rephrase the phrase-repeatability diagnostic (which assumes an ordering between daughters in the tree) in terms *sister-alternability*: the aim of humpback syntax should be to build trees whose sisterhood relations map all and only the alternations attested in themes for that phrase. This recapitulation has the empirical advantage of eliminating the need for transitional phrases, which have the unsatisfactory status of being unrepeatable constituents. Now, transitional phrases can be seen as a side-effect of the unfixed order of sisters in the syntactic structure underlying phrases.

Interestingly, this is a case where the wrong theory can yield the right results most of the time: the phrase-as-alternation and phrase-as-repeating-string models give the same results for all non-edge cases: repeating *ab* and *ba* look identical to alternating *a* and *b* if you ignore the first and last members of the sequence. The set of alternating strings is in fact a superset of the set of repeating ones — and to the extent that there are statistical biases toward one element appearing

at one edge of a phrase, it is in many cases a very slight superset, since the transcriber can fix precedence in alignment with which daughters are most frequently at the left or right edge.

#### 4.4 A derivation for themes

I propose a two-part derivation for humpback themes. The first is the narrow syntax, which recursively builds combinatorial structure via the binary set-forming operation Merge (Chomsky, 1999). The invocation of Merge is crucial here, since Merge gives as its output unordered sets, and thus the structures it builds do not encode precedence relations between sisters in the tree. As shown in 4.2, this neutrality is a desideratum. The second layer of the derivation is a spellout process that works top-down and recursively to convert these trees into strings by alternating between the daughters of each node.

##### I. $\text{Merge}(\alpha, \beta) = \{\alpha, \beta\}$ .

That is, Merge forms an unordered binary set of its two arguments. It can apply to atomic units “*first Merge*”, or to sets created by a previous application of Merge (combining phrases into larger phrases). Below I derive the structure motivated in 4.1 via Merge, yielding the tree in the bottom-right cell.

Order	Merge	Tree
First Merge	$\text{Merge}(a, b) = \{a, b\}$	$\begin{array}{c} \{a, b\} \\ \wedge \\ a \quad b \end{array}$
First Merge	$\text{Merge}(c, d) = \{c, d\}$	$\begin{array}{c} \{c, d\} \\ \wedge \\ c \quad d \end{array}$
Second Merge	$\text{Merge}(\{c, d\}, e) = \{\{c, d\}, e\}$	$\begin{array}{c} \{\{c, d\}, e\} \\ \wedge \\ \{c, d\} \quad e \\ \wedge \\ c \quad d \end{array}$

Third Merge	$\text{Merge}(\{a,b\}, \{\{c,d\}, e\}) = \{\{a,b\}, \{\{c,d\}, e\}\}$	<pre> graph TD     Root["{{a,b}, {{c,d}, e}}"] --&gt; Node1["{a,b}"]     Root --&gt; Node2["{{c,d}, e}"]     Node1 --&gt; a["a"]     Node1 --&gt; b["b"]     Node2 --&gt; Node3["{c,d}"]     Node2 --&gt; e["e"]     Node3 --&gt; c["c"]     Node3 --&gt; d["d"] </pre>
-------------	---	---

Table 3. A derivation of the tree in fig. 8 using Merge.

While the difference between the final tree and the one in Fig. 8 might seem purely notational, Berwick and Chomsky (2015, p. 113) point out an important difference between traditional phrase-structure trees and trees built by Merge: while the former encode precedence (ordering) relations between daughters in the tree, the latter lack this information, and thus are unchanged by rotation of the trees around any of their nodes. That is:

$$(6) \{\{a,b\}, \{\{c,d\}, e\}\} = \{\{\{c,d\}, e\}, \{a,b\}\} = \{\{e, \{c,d\}\}, \{a,b\}\}, \text{ etc.}$$

*whereas*

$$(7) [[[a][b]][[c][d]][e]] \neq [[[[c][d]][e]][[a][b]]] \neq [[e][[c][d]]][[a][b]]], \text{ etc.}$$

**II. Alt(X)** takes as its input a binary set generated by (I) and spells it out as a string that alternates strictly between the set's two members,  $\alpha$  and  $\beta$ . This string is free to begin and end with either of the two elements, so long as one is always followed by the other (or the end of the string). Remember that  $\alpha$  and  $\beta$  can be either atomic units (unary sets), or can themselves be binary sets. Alt applies recursively, so in the latter case each member of the alternation created in the first application of Alt will itself undergo Alt. This process recurs until only terminal nodes (unary sets) are left. For the precise mechanics of this expansion process, see Appendix I, where I derive the first attested theme for C1 using recursive applications of Alt on the final tree in Table 3.

Note that only Merge builds hierarchical structure (nested sets), while Alt uses the hierarchical information encoded at each individual level to build *strings* (which have no hierarchical relations between their members). To account for the repetition of individual units, I assume that all syntactic objects in the derivation, even the terminals, undergo Alt during spellout. The definition above is framed in terms of binary sets, but I propose that alternation applied to a unary set is equivalent to repeating that member. Where X is some set, Alt(X) can be visualized as a cylinder seal with the members of X engraved on its face. When rolled onto clay, it will



produce strings alternating between the members of binary sets, or repeating the members of unary ones.



Figure 10: An Assyrian cylinder seal from 2000 BC that alternates between Gilgamesh and a dragon (without dictating which comes first or last).

For sets with a cardinality of three or greater, the order of the members on the seal will make a difference: strings of *abcabc...* are different from strings of *acbabc...* not only at edges, but all the way through. I will not propose any ternary-branching structures for the songs in my corpus — though neither do I propose that they are not required elsewhere.

In the next section I will test the generalizability of this two-part model across themes corresponding to Phrase C1, by implementing it as a context-free grammar.

## 5. Computational implementations

### 5.1 Context-free grammars (CFGs)

Context-free grammars are a type of formal grammar that can use rewrite rules to generate<sup>4</sup> a set of strings, a *formal language*, corresponding to that grammar. The language being modelled in our case is all the possible themes for a specified humpback phrase — that is, strings that adhere to the core combinatorial structure of the phrase as shown in fig. 8, but may differ in the number of times each alternation is made. This language must contain at least the three C1 themes attested in my small corpus, containing a total of 11 alternations between the two subphrases:

- (8) **a b a c c d c d e c d c d e c d c d e a a b a a c c d e c d c d e**
- (9) **c d c d e c d a a b a a c c d e c d c d e c d c d e a a b a a e c d c d e c d c d e c d c d e**

<sup>4</sup> Or recognize — I will use generative terminology throughout for consistency's sake.

(10) *c d c d e c d c d e c d c d e c d a a b a b a d c d c d e c d c d e c d a a a b a d c d e c d c d e c d c d e*

But the grammar need not generate *only* these strings, just as a grammar of English need not generate only those sentences that have already been said: instead it should be able to generate all *possible* English sentences. In human language, the availability of grammaticality judgements for new sentences makes expanding the language to test new aspects of the grammar easier than in the humpback case: as yet there have been no attempts at eliciting grammaticality judgements from humpbacks. Instead, I will take as my goal to generate all and only the strings adhering to the patterns of alternation attested within themes, since these are robust across themes of the same phrase.

CFGs are composed of an inventory *rewrite rules* that can be used to build hierarchical structure by *expanding* the node (symbol) currently being parsed to one or more other nodes. Every parse must start from a specified “start node”, traditionally labelled S. Here are two possible binary expansions for a start node S:

$S \rightarrow a b$



Figure 11.

$S \rightarrow b a$



Figure 12.

The simple language generated by this grammar is all the strings of length two containing one b and one a (namely, ab and ba). Nodes that cannot be expanded further (like a and b above) are called *terminal nodes*, and are traditionally written in lower case. But the derivation need not terminate after one expansion: the nodes created by the first expansion can themselves be expanded, building trees that are potentially unbounded in depth. Non-terminal nodes (like S) are written in upper case and must either directly or indirectly expand to terminals further down in the tree.

Consider the following one-rule grammar in this relation:

$S \rightarrow a S b$

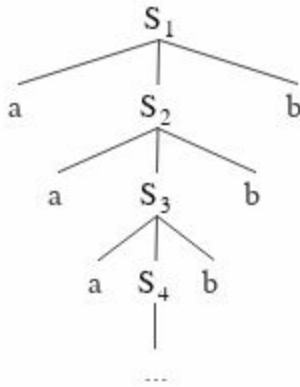


Figure 13. A recursive tree that shows centre-embedding.

This rule is *recursive*, since the left-hand-side is contained within the right-hand-side, allowing for expansions to happen inside other the result of previous expansions.

### 5.1.1 A CFG implementation of Phrase C1

My goal is to implement the Merge-as-alternation system described in 4.4 as a binary-branching context-free grammar. For continuity's sake, I will present a grammar for phrase C1, noting that the model is highly generalizable to other phrases

I should warn the reader that I do not propose that the precise details of this expansion process proposed here have any cognitive reality — only that it does well to approximate with a unitary CFG *the output* of the bipartite expansion process proposed in 4.4. The core hierarchical structure I proposed in Fig. 8 will be built by a small subset of the rules in the grammar; the remaining rules will implement alternation between the daughters of those rules. The two kinds of structure will be interleaved, but always encoded by distinct expansion rules.

Along these lines, the rules in the grammar for this phrase (and others) can be divided into four classes, as laid out below. Only the first of these rule types are specific to a certain phrase, while the remaining three are generated automatically given the rules in (I). The rules in (I) represent the the hierarchically-structured content of the phrase, while the remaining types correspond to Alt, the general mechanism by which that content is converted to strings.

**I. Phrase-structure rules (PSRs):** these build constituent structure. The song node expands to multiple phrases, phrases to subphrases, and subphrases to units. For phrase C1, the following rules yield the structure in fig. 8 (with arbitrary capital letters starting with S used for phrase nodes):

(11)  $S \rightarrow T U$


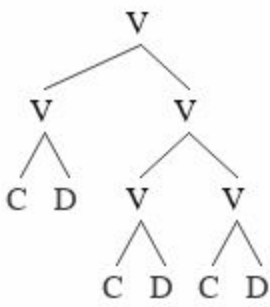
(12)  $T \rightarrow A B$

(13)  $U \rightarrow V E$

(14)  $V \rightarrow C D$

**II. Terminal rules.** These rewrite nonterminal nodes that stand in for units as terminals, which represent the units themselves. The reason for the PSRs not expanding directly to terminal nodes is to allow for intervening expansion by the repetition rules below (III and IV). For phrase C1, these take the form  $A \rightarrow a$  through  $E \rightarrow e$ .

**III. Duplicators.** Any unexpanded nonterminal node may be expanded to two duplicates of itself:  $S \rightarrow S S$ ,  $A \rightarrow A A$ ,  $B \rightarrow B B$ ; etc. These new nonterminals can each either be duplicated again or expand to their daughters via structure-building rules in I (the latter rules out further repetition, and progresses the derivation toward the terminals). The cloning process can happen iteratively, and will take place one less time than the total number of repetitions needed. Below I show how one expansion rule ( $V \rightarrow C D$ ) and one duplication rule ( $V \rightarrow V V$ ) can yield parses for strings of the form  $(CD)^n$ . In the rightmost column I track a metric of the repetitiveness of the expansion to the left, which can be used to make the duplication process probabilistic — for more on this see Appendix IV; I will not introduce probabilities in the body of this dissertation, instead using CFGs as binary acceptors or rejectors of strings (depending on whether or not they can generate them).

Number of duplications	Number of CDs	Parse	Repetitivity = duplicating expansions of V as proportion of total expansions of V
1	2		$1/3 = 0.33$
2	3		$2/5 = 0.4$

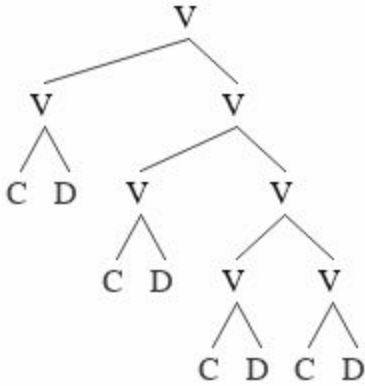
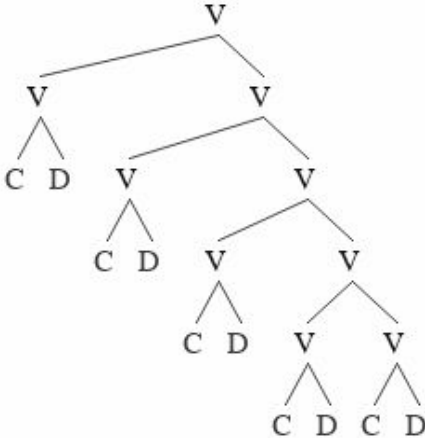
3	4		$3/7=0.43$
4	5		$4/9 = 0.44$

Table 4. Repetitiveness as a balance between duplication and expansion.

**IV. Sesquuplicators<sup>5</sup>:** For a node  $X$  that can be rewritten as  $Y Z$  via expansion rules, these rules rewrite  $X$  as either  $X Y$  or  $Z X$  — that is, a copy of the mother either preceded by its rightmost daughter, or followed by its leftmost. This, in conjunction with the further expansion of  $X$  by duplicators or Core PSRs, has the effect of expanding  $X$  to the unordered set  $\{Y Z\}$ , plus an expansion process whereby unordered sets can be rewritten as *any alternating sequence* of  $Yz$  and  $Zs$ . The sesquuplicators are necessary to the extent that the leftmost and rightmost element in a sequence is variable (see 4.2 and appendix IV). Without these rules, the system would only be able to describe repeating strings — with them it can describe sequences of free alternation.

<sup>5</sup> Substituting the Latin root *sesqui-* “one and a half” for the *du-* in *duplicate*.

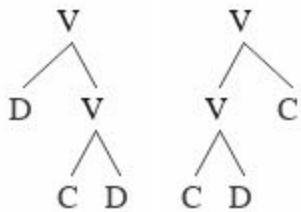


Figure 14.      Figure 15.

The CFG in its entirety is shown in Table 5:

PSRs	Terminal rules	Duplicators	Sesquiplicators
$S \rightarrow T U$	$A \rightarrow a$	$S \rightarrow S S$	$S \rightarrow S T$
$T \rightarrow A B$	$B \rightarrow b$	$T \rightarrow T T$	$T \rightarrow T A$
$U \rightarrow V E$	$C \rightarrow c$	$U \rightarrow U U$	$U \rightarrow U V$
$V \rightarrow C D$	$D \rightarrow d$	$V \rightarrow V V$	$V \rightarrow V C$
	$E \rightarrow e$	$A \rightarrow A A$	
		$B \rightarrow B B$	$S \rightarrow U S$
		$C \rightarrow C C$	$T \rightarrow B T$
		$D \rightarrow D D$	$U \rightarrow E U$
		$E \rightarrow E E$	$V \rightarrow D V$

Table 5. A CFG for generating C1 themes.

## 5.1.2 Generalizability of the grammar

### 5.1.2.1 Undergeneration

To ensure that the grammars do not undergenerate the language, I tested their ability to generate the attested themes corresponding to phrase C1. For this I used an Earley Parser, which, if possible, generates parses given a string of terminals and a CFG. All three themes were given successful parses, showing that they are part of the language described by the grammar. The parses for the three C1 themes are shown below:

First theme:



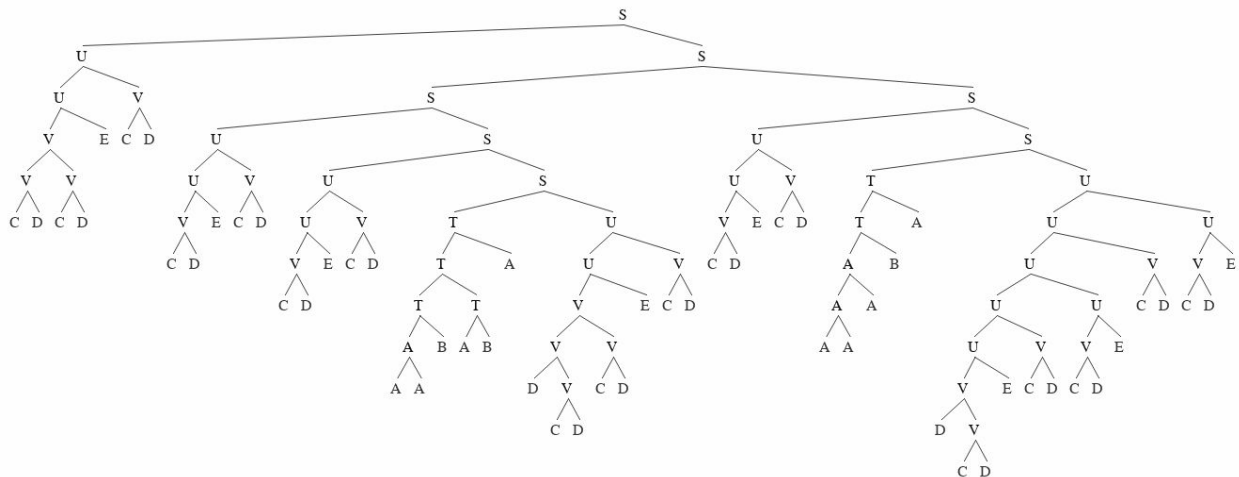


Figure 18. An automatically generated parse of cdcdecdecdecdecdaababadcdecdecdaabadcdecdecdecde.

### 5.1.2.2 Overgeneration

The grammar in Table 5 generates a language a good deal larger than the three attested themes, but it avoids overgeneration in one key respect: namely, it is incapable of repeating non-phrases. This falls out because its only means of implementing repetition, the duplication rules, are unable to expand non-phrases. The grammar was unable to generate, for example, strings that repeated the sequence *ac* at the boundary between *aba* and *cde*. This kind of overgeneration will prove difficult to avoid with simpler models, as discussed in the next two sections. I also found that the grammar was also unable to generate any of 1000 shuffled versions of the first C1 theme (containing all the same letters, but in a randomized order), showing that it does not vastly overgenerate.

### 5.1.3 Pairwise grammar induction

In order to shed light on some of the things the grammar in Table 5 does well, it is revealing to compare it with the automatically-produced rules yielded by *pairwise grammar induction*, an algorithm for automatically building context-free grammars based on statistical properties of the string. The program I used replaces all instances of the most frequent bigram (adjacent pair of terminals) with a new nonterminal, then moves on to the next most frequent pair, makes a rule out of it, and so on iteratively, terminating when no one pair is more frequent than any other.

In the string *abaccdcdecdecdecde*, *cd* is the most common bigram (occurring 9 times), so it will be merged first (rewritten as *F*), yielding the string *abacFeFFeFFe*. The entire induction process for my corpus of C1 themes is included in Appendix III. The process ultimately fails to converge on rules that reduce the three structures to the same start node (indeed, it fails to reduce any of them to a single nonterminal). Below I will lay out two general limitations that prevent the process from performing as well as grammar in Table 5:



1. Groupings of like sequences are specific to the number of repetitions: there is no way of treating *abab* and *ababab* the same for further combination, as it has long been established we must do (the notion of *theme* combines phrases *X* into groupings of  $X^n$ , regardless of the value of  $n$ ).
2. The mergers are *order-dependent* — i.e. the rules encode precedence relations, rather than pure set formation. This means that they are rigid about which of the two daughters should appear at the right and left edges of the theme, and are unable to treat *abab* and *ababa* the same for further grouping.

These challenges reinforce the fact that we will never be able to account for phrase's realization in themes without a model that makes a distinction between phrase structure and linearization.

## 5.2 Finite-state automata

In the Chomsky Hierarchy, which organizes formal languages according to the simplest possible grammar that can generate all and only the strings contained in them, all the simpler language types can be generated by either a simple or a complex grammar — but complex languages can only be generated by grammars of their complexity or greater. The important question with regards to computational complexity is not, then, whether the C1 themes *can* be generated by a CFG (which I have shown they can be), but whether CFGs are the simplest possible computational model — since *ceteris paribus*, we should prefer a less complex model.

The class of languages simpler than the context-free languages are called *regular languages*. The machines required to generate regular languages are called finite-state automata (FSAs, and are composed of a set of states, along with arcs that allow the machine to transition from one state to another, optionally printing a character of the output string at each movement. FSAs have no memory, and so one character in the string can only be dependent on the character immediately preceding it (this is the *First-order Markov assumption*).

Hurford (2012) proposes, in response to Suzuki, Buck, and Tyack's (2006) claims about hierarchical structure (see Section 2.3), that humpback song is in fact describable by a finite-state automaton. I reproduce below his FSA for a song described in P&M (1971):

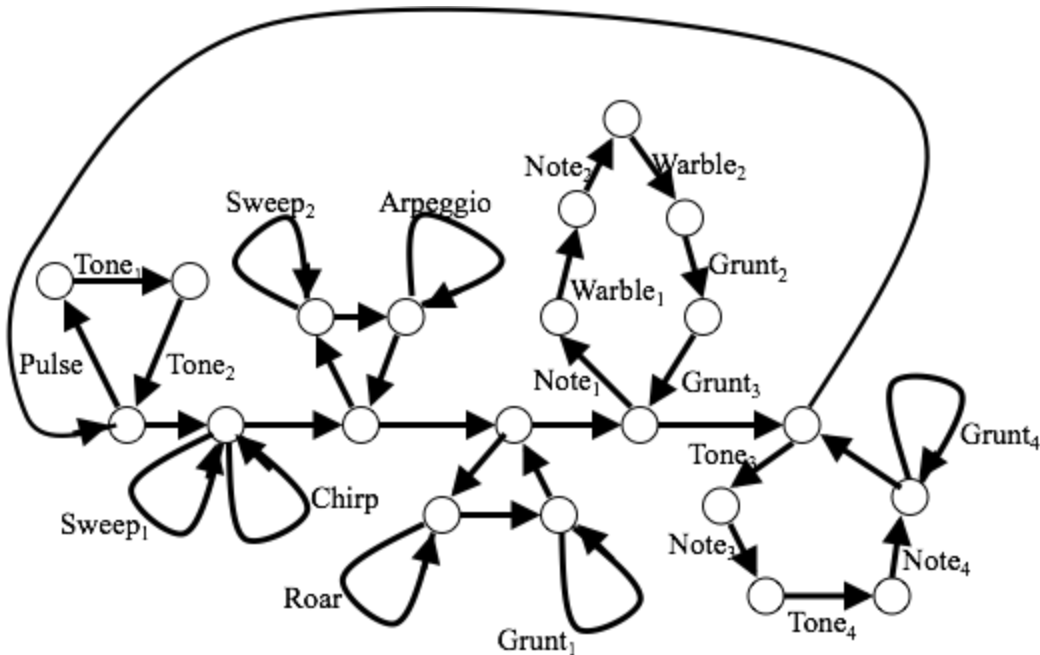


Figure 19. Hurford's (2012) finite-state model of a humpback song reported in Payne and McVay (1971).

But upon inspection of some of the possible routes through this machine, it becomes clear that this model hugely overgenerates: it may be able to generate all the observed songs, but it also generates a vast array of strings that do not adhere to the fundamentals of song structure. The model provides a complete circuit of label-free arcs, and all of the offshoots into phrasal subsystems are actually optional. So technically the empty string (nothing — silence) is included in the language described here. Additionally, from this circuit, one can dip into units from different phrases at will: say, producing a *Sweep* on the first cycle round, a *Roar* on the next, and alternating between these to create strings of the form *Sweep-Roar-Sweep-Roar-Sweep*, etc. In this way the model fails at the task of ruling out unattested alternations, which is the primary explicandum for both Hurford and myself.

The vast overgeneration of this model does not, of course, rule out the existence of some other finite-state model that would better constrain alternation patterns. However, in the footsteps of Chomsky's (1957) pioneering work on English syntax, I will lay out some intuitive reasons why any reasonably parsimonious FSA for humpback themes that does not undergenerate will end up overgenerating, by failing to rule out unattested alternation patterns. Note that this is different from attempting to prove that my corpus of humpback song is context-free, since a finite corpus

of anything (English, French, humpback song) is generable by an FSA<sup>6</sup> — the issue is generating everything *without generating too much*.

Let us take the example of Phrase C1. Low-level alternations (between two units) are easily modeled in finite-state terms, if we treat the alternation between *c* and *d* as a closed system:

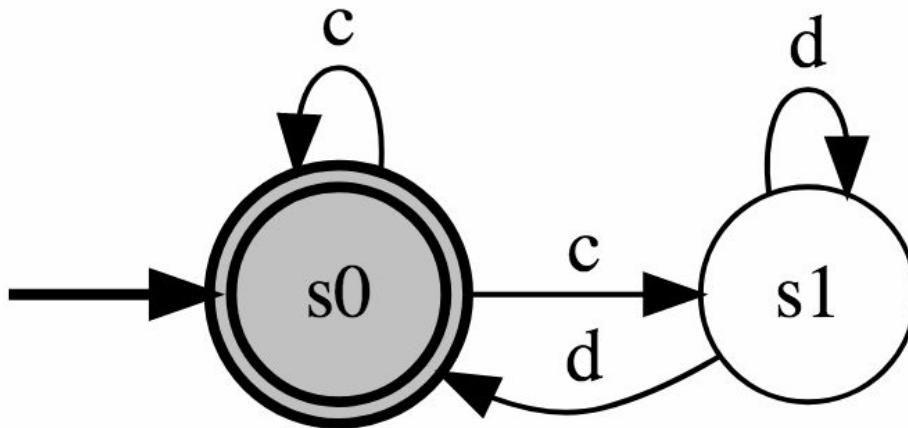


Figure 20. The language recognized by this FSA can be formalized as  $(c^*d^*)^*$ : that is, any string beginning with *c*, ending with *d*, and containing only *cs* and *ds*<sup>7</sup>.

Consider now the following automaton, which fits E into the system:

<sup>6</sup> A simple FSA that can recognize any English sentence would have only have one state, with tens of thousands of arcs looping back to that state and corresponding to every word in the dictionary. But of course the language generated by this FSA is all the strings made up of English words, and will contain inordinately more nonsense than grammatical English sentences.

<sup>7</sup> Also the empty string, since the Kleene operator (\*) designates *zero or more* repetitions of what precedes it.

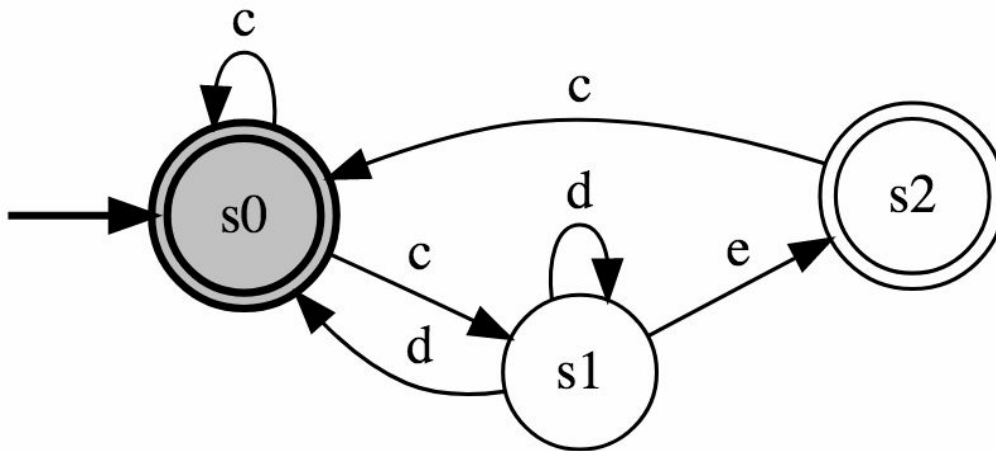


Figure 21. An FSA for the cde subphrase.

In this case, there is no motivation for the corpus for bidirectional transitions between *d* and *e*, or between *e* and *c*, since *c* is never followed by *e*, which in turn is never followed by *d*. Phrases are done away with altogether, and replaced by subsystems with mutual arcs. This also means that the transition between one phrase and the next must be rephrased in terms of a transition between two of their component units. Note, however, that we can now skip out on *d*, and alternate between *c* and *e* (which do not form a constituent). We can fix this by rejigging the transitions as shown below, but even this model will run into problems when we try to model its interaction with *a* and *b*.

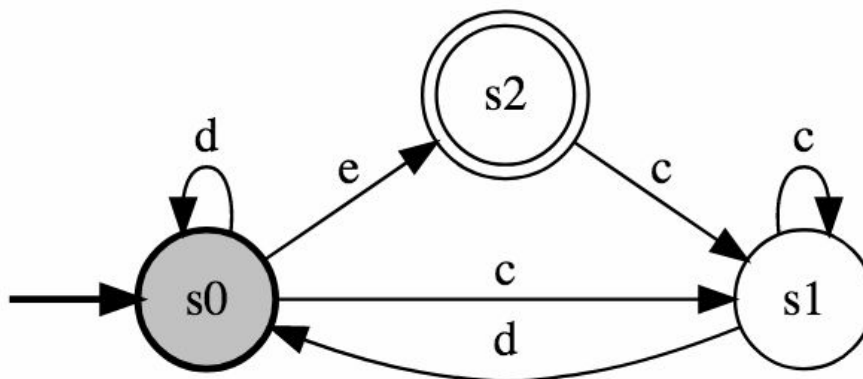


Figure 22. A revised FSA for the cde subphrase.

Recall from 4.2 that the transition can be made from *aba* to any of the three units in *cde*, as shown in the attested strings below (the relevant transitions are bolded):

(17) ... a b **a c c d** ...

(18) ... b a **a e c d** ...

(19) ... a b a d c d ...

We can incorporate *a* and *b* into the FSA by creating two more states, using the FSA can now generate all the transitions in (17) through (19).

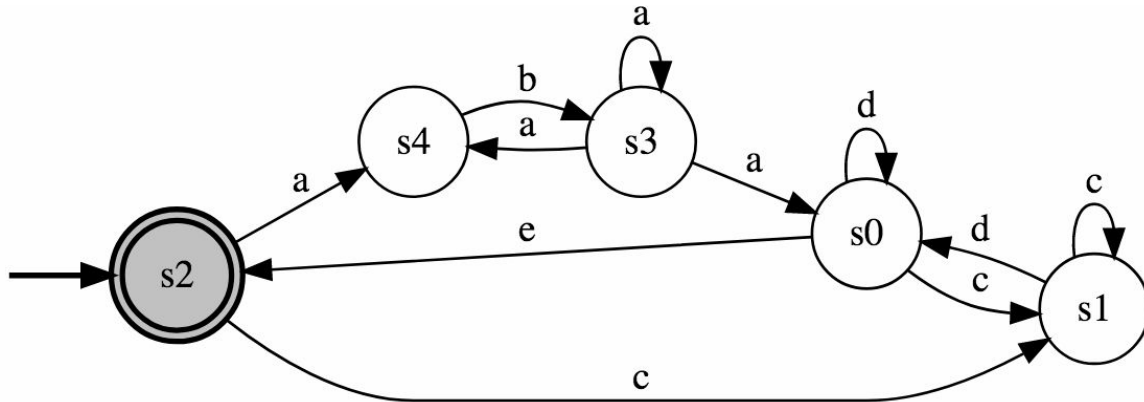


Figure 23. An FSA for generating C1 themes.

Remember that given the goal of generating all and only attested alternation patterns, our model should allow for transitions between any of the terminals of adjoining phrases, but be picky about which of these adjacent pairs can be repeated. There are some unit combinations that appear adjacently in song, in both orders, but *never in alternation*: for example, *aba* can be followed by *e* (as in (18)), and *e* can be followed by *aba* (see (2)), as we alternate in both directions between the AB subphrase and the CDE subphrase; but there is *never* a repeating alternation between *aba* and *e*, (this is encapsulated by the fact that they are not sisters in Fig. 8). This fact is easily captured by a model that has access to hierarchical structure, but turns out to be highly resistant to finite-state modelling, since the amnesia of Markovian systems means that any bidirectional interaction between subsystems will form a loop through which alternation could be made. In this case that loop follows the route  $s2 \rightarrow s4 \rightarrow s3 \rightarrow s0 \rightarrow s2 \dots$ , which prints strings of the form  $(abae)^*$ , even though there is never alternation between *aba* and *e* in the corpus. Thus variability at theme boundaries creates an important overgeneration problem for the kind of simple looping finite-state models of alternation patterns proposed by Hurford (2012).

## 6. Discussion

### 6.1 Self-similarity and Recursion

Like human language syntax, the system proposed in this work is not only hierarchical, but builds embedded structures that are self-similar at multiple levels. This self-similarity, in its most

rudimentary form, can be seen in the observation encapsulated in Fig. 3: songs repeat, and the bits that songs are composed of also repeat. We can now rephrase this and expand this in light of Section 4, and see recursion in humpback song in terms of *alternations between objects that are themselves alternations between other objects*. In all the attested C1 themes, intra-phrasal alternations were always binary (accordingly, the tree in Fig. 8 was strictly-binary branching).

To understand the extent to which humpback syntax is self-similar, we can examine branching structure at various levels. For example, is the sisterhood relation between two unit nodes equivalent to that between two subphrase nodes? It seems from the data discussed in 4.1 that the answer is yes: in both cases (say, in the merger of *c* with *d* and in that of *cde* with *ab*), the combination is characterized by binary alternation between its members. In 4.4, I accounted for the similarity of the system, which exists across three embedded levels, by the recursive application of both Merge and Alt.

What about the level at which the highest-level phrase nodes are dominated by the song node? Here the evidence for binary-branching structure breaks down. Note that in the data in Table 2, the singer never alternates between two phrases before moving on to the next — instead, he cycles between all five in a fixed sequence. This leads us to conclude that the pattern of cycling through multiple phrases seems is best captured by n-ary branching structure (quinary in this case, since there are five phrases.)

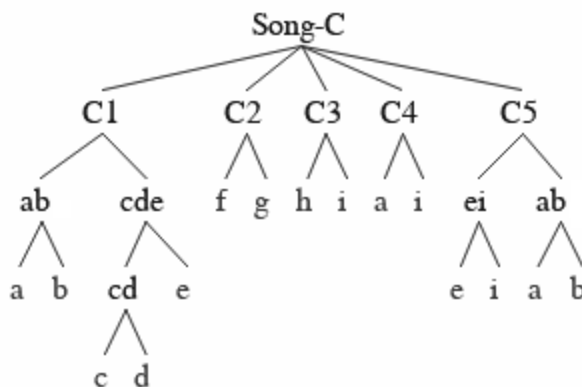

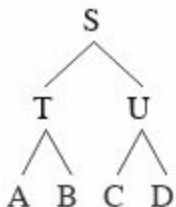
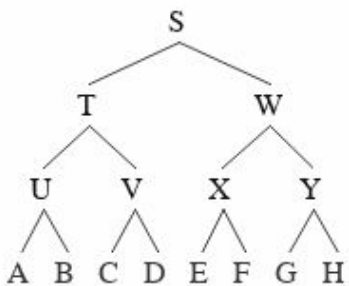
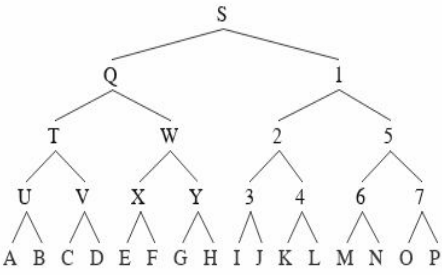


Figure 24. A tree of alternability for Song C.

## 6.2 Limits on depth of embedding

The externalization mechanism for humpback song (Alt) likely impacts the depth of embedding possible in the syntax itself, since the relationship between the number of layers of embedding and the total length of the resulting string, given a constant repetition factor across levels, is much steeper its growth than for human sentences. Below I've plotted this out for idealized symmetric trees up to a depth of 4, using a (conservative) repetition factor of 2 at all levels, and excluding single-unit repetition.

Phrasal layers	Tree	String, with repetition factor of 2 at each level	No. of units involved	String length
0	A	aa	1	2
1	 <pre> graph TD     S --&gt; A     S --&gt; B </pre>	abab	2	4
2	 <pre> graph TD     S --&gt; T     S --&gt; U     T --&gt; A     T --&gt; B     U --&gt; C     U --&gt; D </pre>	ababcedcd ababcedcd	4	16
3	 <pre> graph TD     S --&gt; T     S --&gt; W     T --&gt; U     T --&gt; V     W --&gt; X     W --&gt; Y     U --&gt; A     U --&gt; B     V --&gt; C     V --&gt; D     X --&gt; E     X --&gt; F     Y --&gt; G     Y --&gt; H </pre>	ababcedcdababcedcd efeghghkefeghgh ababcedcdababcedcd efeghghkefeghgh	8	64
4	 <pre> graph TD     S --&gt; Q     S --&gt; 1     Q --&gt; T     Q --&gt; W     1 --&gt; 2     1 --&gt; 5     T --&gt; U     T --&gt; V     W --&gt; X     W --&gt; Y     2 --&gt; 3     2 --&gt; 4     5 --&gt; 6     5 --&gt; 7     U --&gt; A     U --&gt; B     V --&gt; C     V --&gt; D     X --&gt; E     X --&gt; F     Y --&gt; G     Y --&gt; H     3 --&gt; I     3 --&gt; J     4 --&gt; K     4 --&gt; L     6 --&gt; M     6 --&gt; N     7 --&gt; O     7 --&gt; P </pre>	ababcedcdababcedcd efeghghkefeghgh ababcedcdababcedcd efeghghkefeghgh ijijklklijklkl mnmnopopmnmnopop ijijklklijklkl ababcedcdababcedcd efeghghkefeghgh ababcedcdababcedcd efeghghkefeghgh ijijklklijklkl mnmnopopmnmnopop	16	256

		ijijklklijklkl mnmnopopmnmnopop		
--	--	------------------------------------	--	--

Table 6. Exponential growth of string length as a function of depth of embedding.

In human language syntax, the growth of sentence length according to the depth of their embedding is also exponential, but with a base 2 as opposed to a base of 4 for the humpback length. The general formula for string length given  $n$ -arity of branching, a repetition factor of  $x$  and  $L$  layers of phrasal structure, is  $(nx)^L$ . In both human and humpback syntax (so far as has been discussed),  $n$  is fixed at 2. In human syntax,  $x$  is 1, since phrases are not repeated, but in humpback syntax it ranges from 1 to as high as 10 in some cases (see Appendix IV). Thus if humans and humpbacks face similar working memory limitations while computing syntax, it is unsurprising that we see far fewer layers of embedding in humpback song than in human sentences. This limit is an artefact of the linearization method, rather than necessarily a constraint on the depth of the structures humpbacks can build.

### 6.3 Physiological explanations?

Mercado and Handel (2015 — henceforth M&H) propose a two-pronged physiological explanation for song structure that is meant to supplant the need for mental hierarchical structure. The lowest instance of Merge (combining two units that are in alternation) they attribute to the articulatory facts of song production: humpbacks alternate between ingressive and egressive airstream mechanisms while singing (Adam et al., 2013), and M&H propose that bidirectional phonation manifests itself in the alternation between a notes within a phrase. This seems to be a promising route for further inquiry and is consistent with the predictions Adam et al.’s acoustic model, but there is still more work to be done in pinning down the precise acoustic correlates of the ingressive and egressive airstreams.

The second mechanism accounts for the highest level of branching structure, where themes are combined to form the entire song. This they attribute to fluctuations in dive depth: building on the observation that some themes are consistently sung at the time of surfacing, M&H propose that singers’ cycles through themes correlate with their depth. I find this far less plausible than the low-level mechanism, for two reasons. First, it fails to explain the discrete nature of phrase transitions, in response to a continuous variable (depth). Second, if certain themes are tied to certain depths, the order of themes while the whale is diving should be inverted upon their ascent, yielding symmetric patterns of the form A-B-C-D-C-B-A. Instead, themes tends to cycle within songs (A-B-C-D-A-B-C-D): Payne and McVay (1971) first reported this robustness in ordering, both between and across singers. I suspect this higher-level mechanism would be easily falsified by a cursory examination of how dive depths correlates with phrase sequencing.



But crucially, while M&H’s analysis is able to find physiological mechanisms for periodicity at two levels, but would *not* be able to explain four or five. I tried to demonstrate in 4.5 that the repetition patterns in phrases like C1 require *two more layers* than traditionally allowed for. These extra intermediate projections can have no physiological mechanism in M&H’s model, since they generate periodicities between that of the dive cycle and that of the ingressive-egressive alternation. This leads us to assume, in the absence of a plausible physiological explanation, that these structures are cognitive in origin.

## 6.4 Hybridization: an independent constituency test?

Garland et al (2017) discuss a rare but fascinating phenomenon that offers a second lens through which to look at constituency in humpback song (the first being repetition). When a population is transitioning between an old and new song, individuals will occasionally sing *hybrid* songs in which elements from the old and new songs are remixed and woven together. The authors suggest that in some cases the songs alternate between old and new phrases, or else the phrases themselves are hybrid, incorporating both new and old units.

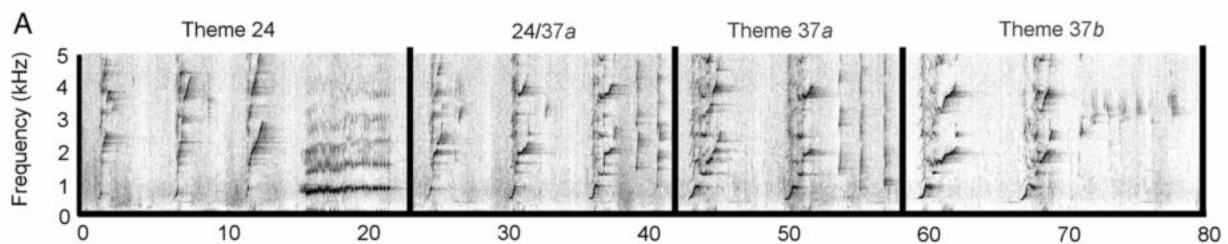


Figure 25. Theme 24 and 37 are from different songs, and are spliced together here. The segment of interest is between roughly 22 and 41 seconds, where the first units to appear in Theme 24 are immediately followed by the second units to appear in Theme 37.

However, note that in the data provided there, there are no *repeating* sequences containing elements from the two songs: the supposed unit recombinations are in “transitional phrases”, which I argued in 4.2 should not be seen as phrases at all, but as an artefact of transcribers caring about order, and singers not. The data does, however, clearly show productive recombination of themes at the song level, reinforcing the stability of the phrase as a syntactic object.

## 6.5 Syntax, Phonology and Beyond

Given the conclusions I have drawn about the nature of hierarchical structure in humpback phrases, I will now discuss a framework for comparing the cognitive capacity underlying humpback song with other cognitive capacities in the animal kingdom, including, but not limited to, human language. A commonly drawn distinction in discussions of the evolution of syntax is

between *phonological syntax* and *lexical syntax* (Marler 1977). In lexical syntax alone does the meaning of a larger signal derive compositionally from the meanings of its components: the meaning of a phrase is a function of the meanings of the lexical items that compose it, whereas the meaning of a lexical item is not a function of the meanings of the phonemes that compose it. Since this distinction uses semantics as a benchmark, it is a functional rather than a formal one, and is neutral regarding the formal characteristics of the computational system. A *formal* distinction between phonology and syntax can be made roughly along the lines of Heinz and Idsardi (2013): that is, between strings and trees. Heinz and Idsardi argue that all phonotactic constraints, including long-distance ones, like the results of harmony processes, are not only regular but subregular — that is, they can be generated by a constrained subset of the regular grammars.

In the human syntax-phonology distinction, it just so happens that these two properties, semantic compositionality and hierarchical structure, correlate. But it need not be that way in other systems. I take the two distinctions to run orthogonally, and I will show that there are attestations of systems exhibiting all four possible permutations, as schematized below:

	Compositional semantics (+cs)	No compositional semantics (-cs)
Generates strings (-h)	<b>Calls of Campbell’s monkeys</b> (Schlenker, Chemla & Zuberbühler, 2016); <b>Word concatenations in early child language</b> (Roy, Copley & McCune, 2015)	<b>Human phonology</b> (Heinz & Idsardi, 2013); <b>Birdsong</b> (Gentner et al, 2006 vs. Reuland, 2013)
Generates trees (+h)	<b>Human language syntax</b>	<b>Human music</b> (Lerdahl & Jackendoff, 1983; Katz & Pesetsky 2009); <b>Humpback song?</b>

Table 7. The four permutations of the binary features  $\pm cs$  (compositional semantics) and  $\pm h$  (hierarchical structure)

Note: I use “strings” here as judged by the doctrine of “strings until proven trees” — that is, as the null hypothesis for questions of hierarchical structure, rather than a positive proposal that they have no internal structure.

**Quadrant I** [-h, -cs] systems perform computations without any memory — and fall in the class of regular grammars in the Chomsky hierarchy.

Bird song is a point of controversy here. Gentner et al, 2006 offer starlings' ability to distinguish between sequences of the form  $A^n B^n$  and sequences of the form  $(AB)^n$  as evidence for their ability to learn context-free patterns. But Reuland (2013) points out that, especially at small coefficients, there are finite-state representations for such sequences. He argues that it does not follow from the starlings' ability to differentiate between the two patterns that they do so using multiply-embedded structures.

As outlined in 5.2, a finite corpus of any language can be modelled using a finite-state automaton. But beginning with Chomsky (1957), it has been clear that any empirically adequate and reasonably parsimonious model of human language syntax will have to be at least context-free — that is, deal in trees, not strings. Additionally, it seems that syntax produces surface patterns that approach context-freeness — that is, they would be context free if repeated to infinity (e.g. centre-embedding, or increasingly long intervals between dependent elements, Carnie 2010). The hierarchical nature of human syntax has been supported by a half-century long tradition of testing the validity of posited structures against novel sentences. While syntacticians often vehemently disagree about the correct constituent structure, almost all accept that constituent structures exist (see Dryer 2009 for one exception).

I have argued that we have a reliable constituency test for humpback syntax that leads us to multiply-embedded structures, as in the case of human language — and moreover, one that does not rely on native-speaker grammaticality judgements or semantics, but is apparent in the string. I showed in 5.2 that finite-state models like the one proposed by Hurford (2012) may be parsimonious, but are not empirically adequate, suggesting that these patterns are resistant to being modelled without reference to hierarchy.

**Quadrant II** [-h,+cs] systems are necessarily limited in their compositionality<sup>8</sup> — both examples given in Table 7 are able to combine at most a handful of items — but demonstrably compositional nonetheless.

There is at least one piece of compelling evidence for compositional concatenation in nonhuman animals. Schlenker, Chemla & Zuberbühler's (2016) work on Campbell monkey calls showed a pattern of context-dependence with respect to the calls that led them to posit call-internal structure. The suffix that the authors call *-oo* appeared to be freely combinable with calls

---

<sup>8</sup> Indeed, it is difficult to conceive of a system that generates large compositional structures *without* hierarchy — especially a system capable of discrete infinity (or discrete very-very-bigness: Pinker (1994) estimates the number of *meaningful* English sentences of length  $n$  is around  $10^n$ .)

denoting specific predators, apparently contributing a meaning equivalent to the English suffixes *-ish* or *-like*: monkeys called “eagle” when an eagle was present, but called “eagle-ish” for a generic danger from above.

**Quadrant III** [+h, -cs] is occupied by human language alone, in the opinion of many evolutionary linguists (Berwick and Chomsky, 2015; Bolhuis et al, 2014, *inter alios*). Although recursion yields the structures necessary for compositional semantics, Berwick and Chomsky propose that the emergence of the former need not have brought on the latter. I return to this point in 6.6.

**Quadrant IV** [+h, -cs] is where I argue humpback song should be placed, given the argument presented here in support of the need for hierarchical structure — again, with the assumption of “asemantic until proven semantic”.

Also of interest for this quadrant is work attempting to apply a generative framework to the syntax Western Tonal music. This line of inquiry dates back to Lerdahl and Jackendoff’s *Generative Theory of Tonal Music* (1983). The branching structures they propose map tone stability, a hierarchy of the chords in a given scale: the most stable chord in a key is the tonic (the first note in the scale), followed by the dominant (the fifth), then the subdominant (the fourth), and so on. Moving to less stable chords builds tension, while moving to more stable chords gives resolution. These patterns of tension and resolution exist at multiple, nested levels.

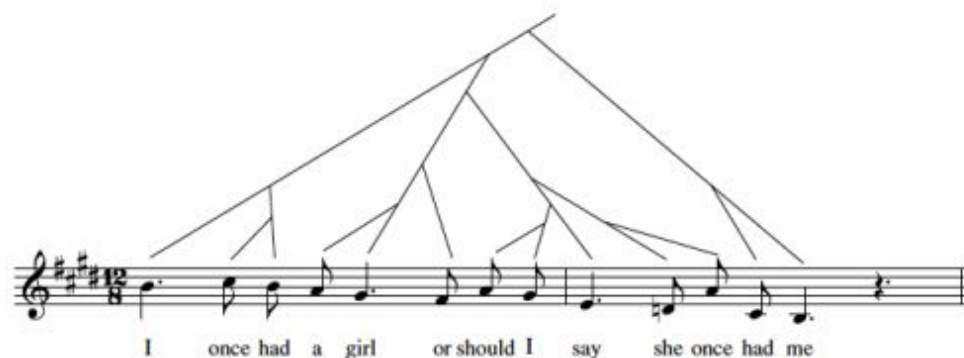


Figure 26. The pitch structure of the opening line of “Norwegian Wood”, according to Lerdahl & Jackendoff (1983, p. 50). Lines of projection are signified by straight lines, and lines that end by merging with a longer straight line form complements. Thus the note coinciding with “I” is the head of the entire phrase, because it is the most stable note in it (the tonic — the first note in the scale).

More recently, Katz & Pesetsky (2009) have proposed that the syntaxes of language and music are not only similar but identical. All that differs, in their view, is the objects being manipulated — this is consistent with humpback song, as I will discuss in the next section.

## 6.6 The evolutionary picture

Many theories of evolutionary linguistics place special emphasis on the human ability to process multiply-embedded hierarchical structures. *Why Only Us* (Berwick & Chomsky, 2015) takes an extreme position in this regard. Berwick & Chomsky (henceforth B&C) argue that evolutionary linguist should aim to provide the simplest possible story for human language evolution. B&C converge on this point of view using evidence from both generative syntax and the evolutionary record, which shows that language emerged extremely recently — between 200,000 and 60,000 years ago (p. 110) — and very quickly. This simplicity, in their view, means excluding from the question of what evolved everything but the *Basic Property* of language. The Basic Property for them is the fact that “each language provides an unbounded array of hierarchically structured expressions”, and the simplest possible explanation for this is Merge (as defined in, e.g., Chomsky, 1999).

If humpback songs are generated by Merge, I take this correspondence with humans to be due to convergent evolution, since the ability to learn or spontaneously produce deeply embedded structures is absent both in other primates and, so far as we know, in other cetaceans. The cognitive backdrop in which humpback Merge would have evolved is entirely different: it does not, as far as we know, interface with semantics, and phrase structure is only rarely used as a means of generating new permutations (as we saw in Section 6.4). Despite these functional differences, if B&C’s proposal that Merge’s evolution was both extremely simple and alinguistic is correct, then the evolution of Merge in a different species and cognitive backdrop is not surprising. Merge on its own does not yield the layman’s “language” (strings of consecutive words or signs that mean things) — it yields recursive binary set-formation. In the human case, this has been harnessed for compositional semantics, but it need not have been in the humpback case.

Collier et al (2014) present a compelling case that complex words and phonology need not have preceded syntax, since some limited form of syntax is attested elsewhere in the animal kingdom (call combination, e.g. the work by Schlenker et al in Section 6.5). While I, along with most generative biolinguists, would not consider such examples syntax in the meaningful sense, this seems a plausible proposal. Collier et al point out that phonology would only be needed as a way of expanding lexicon once the need for new lexical items surpassed the number of discriminable contrasts between atomic sounds.

It is conceivable, then, that humpback syntax is underlain by a cognitive system similar to human syntax, but that the objects under manipulation are something other than lexical items (the atoms of thought). In this way it could prove highly similar to human music. However, the findings

presented here do not bear on humpback syntax's interface with meaning (Logical Form, or LF). They do, however, provide a model of its interface with Phonetic Form (PF) that paints a very alien picture: for all the diversity of linear order in human language syntaxes, there is nothing that even approaches the nested repetition patterns of humpback songs.

## 7. Future work

In the vein of 6.6, one line of inquiry would be to begin the search for four or more phrasal layers in other humpback songs currently recorded — though if my reasoning in 6.1 is sound, this is unlikely to return a positive result.

One more promising direction would be to look at the non-song vocalizations of humpbacks, with an eye to seeing if there reason to posit hierarchical structure there as well. In addition to singing songs, humpbacks (both male and female) make other concatenations of sound units that lack this repetitive structure, are not stereotyped, and vary from context to context. These “bouts” of calls are almost certainly semantically contentful, and could be compositional: Noad et al (2015) investigated sequences of calls made within 3.9 seconds from one another, and found very little repetition of entire bouts (unlike songs, which are highly stereotyped). The sequences were non-random, but also not entirely predictable by transitional probabilities, making their information-bearing potential high (though still low compared to that of human speech, if measured against time).

Unfortunately the call bouts, unlike songs, lack a clear diagnostic for phrasehood, since there are no internal repetition patterns. This does not confirm or disprove the existence of hierarchical structure within the bouts, but does suggest that any hierarchical structure that does exist is not externalized in the same way as it is in the repetition patterns of male songs.

So far, most of the work on call bouts (as well as much of the work on songs) has been anchored in information theory. While information theory has a great deal to teach us about the *possibility* of compositional semantics in these sequences, the exact nature of the semantics seems unlikely to be discovered without something akin to linguistic fieldwork — an unwieldy task given the physical and cognitive divides between our two species. Humpback social sounds are no low-hanging fruit, but promise a hugely exciting area of inquiry for the next generation of zoosyntacticians.

## Acknowledgements

I would like to thank Ellen Garland, Luke Rendell and Luca Lamoni for welcoming me on as an intern in their lab at St. Andrews in the summer of 2016, and Ellen in particular for teaching me the ropes of song transcription. I developed many of the ideas in this dissertation while staring at spectrograms all day in their lab. I also thank Ellen Garland and Michael Poole for allowing me to use their data here (Song A — which is discussed in Appendix IV). I am also very grateful Paul Knapp, Jr., who gave me permission to use the Song C data here, which contained the most interesting themes for phrase-internal structure.

My Director of Studies, Paula Buttery, was enthusiastic about this project from the start, and enormously helpful with the computational work. She took the time to write me some code (a parser and the grammar inducer used in 5.1.3), and show me how to use it. I would not have figured any of this out if not for her. Arik Kershenbaum also helped me figure out some helpful software for PCFGs, and gave feedback to me in the early stages of this dissertation.

I am extremely grateful to my supervisor, Theresa Biberauer, who gave copious and insightful comments which I have addressed to the best of my ability. She helped me clarify my discussion and think about explaining my reasoning to a readership beyond myself, which is especially vital when writing at the intersection of two largely non-interactive fields.

Lastly, I'd like to thank my family and friends who let me blab about this stuff to them over the past few years, shared in my enthusiasm for humpbacks, and encouraged me to pursue this for my dissertation despite the wackiness (and non-humanness) of the subject matter. I'm grateful to my dad for challenging me to be clear, and to think about the bigger picture, as someone who's a sucker for the formal details.

## Appendix I: Derivation with Alt(X)

Here I derive the first theme of C1 (given in Table 2) from the structure in Fig. 8, by recursively applying Alt(X) (Section 4.4) to all the sets in the structure. In practical terms, this will mean replacing all sets  $\{\alpha, \beta\}$  by Alt( $\alpha, \beta$ ), including sets embedded within other sets<sup>9</sup>. Alt(X) is then probabilistically reduced to a concatenation — Concat(Y) — where Y is some string of alternating  $\alpha$ s and  $\beta$ s. This happens iteratively, from the outside in, until all instances of Alt have

---

<sup>9</sup> Alt(X) should apply to unary sets too (the individual terminals), but this would have made the derivation laughably complex (perhaps it already is), so I've omitted this and duplicated individual units by fiat (i.e. allowing the alternations created by Alt(X) not to be strict).

been reduced to concatenations, giving the linearization. Our input is the following nested structure:

$$\{\{a,b\},\{\{c,d\},e\}\}$$

The derivation for *abaccdcdcdcdcdcddeaabaaccdcdcd* is as follows (newly introduced concatenations on each line are in **bold**):

$$\begin{aligned} & \text{Alt}(\{\{a,b\},\{\{c,d\},e\}\}) \\ &= \text{Alt}(\text{Alt}(a,b), \text{Alt}(\text{Alt}(c,d),e)) \\ &= \text{Concat}(\text{Alt}(a,b), \text{Alt}(\text{Alt}(c,d),e), \text{Alt}(a,b), \text{Alt}(\text{Alt}(c,d),e)) \\ &= \text{Concat}(\text{Concat}(a,b,a), \text{Alt}(\text{Alt}(c,d),e), \text{Concat}(a,a,b,a,a), \text{Alt}(\text{Alt}(c,d),e)) \\ &= \text{Concat}(\text{Concat}(a,b,a), \text{Concat}(\text{Alt}(c,d),e, \text{Alt}(c,d),e), \text{Concat}(a,a,b,a,a), \text{Concat}(\text{Alt}(c,d),e, \text{Alt}(c,d),e)) \\ &= \text{Concat}(\text{Concat}(a,b,a), \text{Concat}(\text{Concat}(c,c,d,c,d),e), \text{Concat}(\text{Concat}(c,d,c,d),e), \text{Concat}(a,a,b,a,a), \\ & \quad \text{Concat}(\text{Concat}(c,c,d),e, \text{Concat}(c,d,c,d),e)) \\ & \dots \\ &= \text{Concat}(a,b,a,c,c,d,c,d,e,c,d,c,d,e,c,d,c,d,e,a,a,b,a,a,c,c,d,e,c,d,c,d,e) \end{aligned}$$

## Appendix II: CFGs to PCFGs

### Probabilistic context-free grammars (PCFGs)

Probabilistic context-free grammars (PCFGs) are CFGs that introduce a bias toward some rules over others in the form of probabilities assigned to each rule in the grammar, such that the probabilities for all the rules with a given left-hand side sum to 1. This has the effect of preferring some strings in the language over others, and introducing a gradient notion of grammaticality alongside the categorical one introduced by generability.

Using the example in 5.1, if we give the first rule a higher probability than the second, as shown below, the two strings in the language will both still be generable, but one will be more likely than the other, in proportion to the likelihood of the path taken to get there from the start node.

(15)  $S \rightarrow a b [0.9]$

(16)  $S \rightarrow b a [0.1]$

Given this grammar, we would expect a corpus of this language to contain nine *ab* strings for every *ba* string. For grammars with many-layered derivations, the probability of the entire



derivation can be calculated as the product of all the rules used in the derivation, or alternatively, the sum of their log probabilities (so as to avoid the incredibly small numbers resulting from multiplying several near-zero values together).

We can *train* CFGs by assigning probabilities to their rules, turning them into PCFGs. Usually this will be done in a way that maximizes the overall probabilities of parses for a corpus of attested strings.

## Training

I trained the PCFGs using Mark Johnson’s (2013) *C implementation of the Inside-Outside algorithm for estimating PCFGs given a corpus of terminal strings* (downloaded from his website — see Software). This software takes as its input a CFG and a corpus of strings, and “trains” the CFG — assigns probabilities to the rules — so as to maximize the overall probabilities of the parses for the strings in the corpus. In this way, the grammar is fitted to that corpus, and it creates a probability distribution for the variable aspects of the phrase’s structure. For instance, the balance between the repetition rules and expansion rules for a given left-hand-side will encode the mean repetitiveness of that phrase, in the way outlined in table 4. Similarly, the preference for one daughter or another at the left or right edge of the alternation will be encoded in the use or disuse of the respective sequiplicator rules.

There is one interesting deficit in the way repetitiveness is modelled in table 4. Note that the probabilities approach an asymptote at  $\frac{1}{2}$ , rather than increasing linearly with increasing numbers of repetitions. I suspect this might prove an issue if the grammar had to generate a corpus of novel songs, since if both the initial clones clone again (rather than one capping off to CD), Pandora’s Box is open, and it becomes quite likely that the expansion will go on infinitely. However, the exact probabilistic implementation was not a key part of this project, where simple untrained CFGs were sufficient, since I was interested in binary acceptance or rejection of strings, rather than the probabilities assigned to them.

## Appendix III: Induction

```
a b a c c d c d e c d c d e c d c d e a a b a a c c d e c d c d e
c d c d e c d a a b a a c c d e c d c d e c d c d e a a b a a e c d c d e c d c d e c d c d e
c d c d e c d c d e c d c d e c d a a b a b a d c d c d e c d c d e c d a a b a d c d e c d c d e c d c d
e
```

New Rule:  $c d \rightarrow F \text{---} 40$

```
a b a c F F e F F e F F e a a b a a c F e F F e
```

FFeFaabaacFeFFeFFeaaabaeFFeFFeFFe  
 FFeFFeFFeFaababadFFeFFeFaaabadFeFFeFFe

New Rule:  $F e \rightarrow G$ ---20

abacFGFGFGaabaacFGG  
 FGFaabaacFGFGFGaabaacFGFGFG  
 FGFGFGFaababadFGFGFGaabadFGFGFG

New Rule:  $F G \rightarrow H$ ---17

abacHHHaabaacGH  
 HFaabaacGHHaabaacHHH  
 HHHFaababadHHFaabadGHH

New Rule:  $H H \rightarrow I$ ---9

abacIHaabaacGH  
 HFaabaacGIaabaacIH  
 IHFaababadIFaabadGI

New Rule:  $a a \rightarrow J$ ---9

abacIHJbJcGH  
 HFJbJcGIJbJeIH  
 IHFJbabadIFJabadGI

New Rule:  $J b \rightarrow K$ ---4

abacIHKJcGH  
 HFKJcGIKJeIH  
 IHFKabadIFJabadGI

New Rule:  $a b \rightarrow L$ ---3

LacIHKJcGH  
 HFKJcGIKJeIH  
 IHFKLadIFJLadGI

New Rule:  $K J \rightarrow M$ ---3

LacIHM cGH  
 HFMcGIMeIH  
 IHFKLadIFJLadGI

New Rule:  $L a \rightarrow N$ ---3

N c I H M c G H  
 H F M c G I M e I H  
 I H F K N d I F J N d G I

New Rule: I H → O ---3  
 N c O M c G H  
 H F M c G I M e O  
 O F K N d I F J N d G I

New Rule: c G → P ---2  
 N c O M P H  
 H F M P I M e O  
 O F K N d I F J N d G I

New Rule: N d → Q ---2  
 N c O M P H  
 H F M P I M e O  
 O F K Q I F J Q G I

New Rule: M P → R ---2  
 N c O R H  
 H F R I M e O  
 O F K Q I F J Q G I

## Appendix IV: Phrase A2

While I will use phrase C1 throughout the body of this dissertation, I have made some progress in generalizing this analysis to other phrases. I had access to a second corpus consisting of transcriptions of four recordings made off the coast of Mo'orea in French Polynesia in 2014, and transcribed by me in the summer of 2015. I use them here with the kind permission of Ellen Garland and Michael Poole, the joint owners of the data. The Polynesian dataset contains 5428 units, and lasts 1h48m.

Three phrases in the corpora, one from Song C and two from Song A, exhibit two layers of within-phrase alternation (the simplest phrases have one; phrase C1 had three.) is one such phrases. A typical attestation is as follows:

(17) hifgfgfgfgfgfgfgfghihifgfgfgfgfgfgfghihifgfgfgfgfgfgfghihifgfgfgfgfgfgf



All tree diagrams were generated using:

Eisenbach, M. and Eisenback, A. 2016. “phpSyntaxTree v1.12”.

<http://ironcreek.net/phpsyntaxtree/>

Unless otherwise stated, all finite-state automata were generated using:

Zuzak, I. and Jankovic, V. “FSM Simulator”. Built with Noam, Bootstrap, Viz.js, and jQuery.

[http://ivanzuzak.info/noam/webapps/fsm\\_simulator/](http://ivanzuzak.info/noam/webapps/fsm_simulator/)

## Bibliography

- Adam, O., Cazau, D., Gandilhon, N., Fabre, B., Laitman, J. T., & Reidenberg, J. S. (2013). New acoustic model for humpback whale sound production. *Applied Acoustics*, 74(10), 1182–1190. <https://doi.org/10.1016/j.apacoust.2013.04.007>
- Berwick, R. C. and Chomsky, N. (2015). *Why Only Us: Language and Evolution*. Cambridge, MA: The MIT Press.
- Bolhuis, J. J., Tattersall, I., Chomsky, N., & Berwick, R. C. (2014). How Could Language Have Evolved? *PLoS Biology*, 12(8), 1–6. <https://doi.org/10.1371/journal.pbio.1001934>
- Carnie, A. (2010). *Constituent Structure*. Oxford: OUP
- Cholewiak, D. M., Sousa-Lima, R. S., & Cerchio, S. (2013). Humpback whale song hierarchical structure: Historical context and discussion of current classification issues. *Marine Mammal Science*, 29(3), 312–332. <https://doi.org/10.1111/mms.12005>
- Chomsky, N. (1957). *Syntactic Structures*. The Hague: Mouton De Gruyter.
- Chomsky, N. (1999). *Derivation by phase*. Cambridge, MA: The MIT Press.
- Chomsky, N. (2007). Approaching UG from below. In: Uli Saurland and Hans-Martin Gartner (eds.), *Interfaces + recursion = language?: Chomsky's Minimalism and the view from syntax-semantics*, 1-29. Mouton de Gruyter.
- Chomsky, N. (2013). “What Is Language” (First lecture of Dewey Lectures 2013: “What Kind of Creatures Are We?” *The Journal of Philosophy* 110(12): 645–662.
- Dryer M.S. (2009) The Branching Direction Theory of Word Order Correlations Revisited. In: Scalise S., Magni E., Bisetto A. (eds) *Universals of Language Today*. Studies in Natural Language and Linguistic Theory, vol 76. Dordrecht: Springer.
- Frazer, L. N., & Mercado, E. (2000). Sonar model for humpback whale song. *IEEE Journal of Oceanic Engineering*, 25(1), 160–182. <https://doi.org/10.1109/48.820748>
- Garland, E. C., Goldizen, A. W., Rekdahl, M. L., Constantine, R., Garrigue, C., Hauser, N. D., ... Noad, M. J. (2011). Report Dynamic Horizontal Cultural Transmission of Humpback Whale Song at the Ocean Basin Scale. *Current Biology*, 21(8), 687–691. <https://doi.org/10.1016/j.cub.2011.03.019>

- Garland, E. C., Rendell, L., Poole, M. M., & Noad, M. J. (2017). Song hybridization events during revolutionary song change provide insights into cultural transmission in humpback whales. *Proceedings of the National Academy of Science*, 140(4), 7822–7829. <https://doi.org/10.1121/1.4970982>
- Gentner, T. Q., Fenn, K. M., Margoliash, D., & Nusbaum, H. C. (2006). Recursive syntactic pattern learning by songbirds. *Nature*, 440(7088), 1204–1207. <https://doi.org/10.1038/nature04675>
- Heinz, J., & Idsardi, W. (2013). What Complexity Differences Reveal About Domains in Language. *Topics in Cognitive Science*, 5(1), 111–131. <https://doi.org/10.1111/tops.12000>
- Hurford, J. (2012). *The Origins of Grammar*. Oxford: Oxford University Press. Section 1.3.3: “Hierarchically Structured Behaviour”.
- Katz, J., & Pesetsky, D. (2009). The identity thesis for language and music. Draft Published Online.; [lingBuzz/000959](https://lingbuzz.000959), (January). <https://doi.org/10.1525/mts.1985.7.1.02a00120>
- Kayne, Richard S. (1994). *The Antisymmetry of Syntax*. Linguistic Inquiry Monograph Twenty-Five. MIT Press.
- Ladd, R. (1986). Intonational phrasing: the case for recursive prosodic structure. *Phonology*, 3(1986), 311. <https://doi.org/10.1017/S0952675700000671>
- Lerdahl, Fred, and Ray Jackendoff. (1983). *A generative theory of tonal music*. Cambridge, MA: MIT Press.
- Marler, P. (1977). *The structure of animal communication sounds*. Recognition of Complex Acoustic Signals (T. Bullock, Ed.). Berlin: Dahlem Konferenzen.
- Mercado, E., & Handel, S. (2015). Understanding the structure of humpback whale songs (L). *The Journal of the Acoustical Society of America*, 2947(L). <https://doi.org/10.1121/1.4757643>
- Noad, M. J., Rekdahl, M. L., Dunlop, R. A., & Garland, E. C. (2015). Non-song social call bouts of migrating humpback whales, 3042. <https://doi.org/10.1121/1.4921280>
- Payne, R. S., & McVay, S. (1971). Songs of humpback whales. *Science*, 173(3997), 585–597. <https://doi.org/10.1126/science.173.3997.585>
- Pinker, Steven. (1994). *The Language Instinct*. New York: William and Morrow. Chapter 4.
- Reuland, S. (2013). “Recursivity of Language: What Can Birds Tell Us about It?” In Bolhuis, J. and Everaert, M (Eds). *Birdsong, Speech and Language: Exploring the Evolution of Mind and Brain*. Cambridge, MA: The MIT Press.
- Roy, I., Copley, B., & McCune, L. (2015). The development of Merge and Recursion as generative processes in the transition from single words to sentences. Unpublished draft.
- Sabrina Gerth (2015). *Memory Limitations in Sentence Comprehension: A Structural-based Complexity Metric of Processing Difficulty*. Universitätsverlag Potsdam. pp. 87–. ISBN 978-3-86956-321-3.
- Schlenker, P., Chemla, E., & Zuberbühler, K. (2016). What Do Monkey Calls Mean? *Trends in Cognitive Sciences*, 20(12), 894–904. <https://doi.org/10.1016/j.tics.2016.10.004>

Suzuki, R., Buck, J. R., & Tyack, P. L. (2006). Information entropy of humpback whale songs. *The Journal of the Acoustical Society of America*, 119(3), 1849–1866.  
<https://doi.org/10.1121/1.2161827>